Contador Controlado por UART

Roberto Federico Farfán

Objetivo: El objetivo de este trabajo es implementar un contador de 12 bits controlado por UART. El sistema digital se implementará en una FPGA Altera Cyclone II EP2C5T144, mediante código VHDL. La lógica se divide en bloques, los cuales se desarrollaron en las clases de Circuitos Lógicos Programables, perteneciente al posgrado de Sistema Embebidos de la UBA. En el trabajo se simulan los bloques en las distintas etapas del desarrollo, utilizando software como GHDL y GTKwave.

Introducción

El contador se implementó utilizando una FPGA Altera Cyclone II EP2C5T144, dos cd4050, un conversor de TTL a UART USB-TTL-PL2303, doce resistencias y doce diodos led.

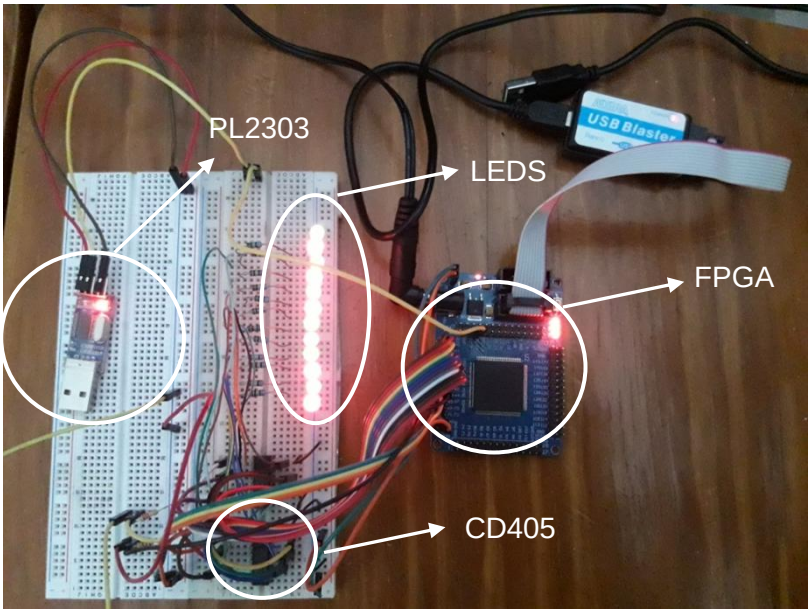

Figura 1. Circuito armado.

El contador controlado por UART recibe caracteres enviados desde una PC por medio del USB-TTL-PL2303. El contador se diseño de forma que al recibir el carácter I (inicial de la palabra incremento), sumará una unidad en el contador. Por otro lado, cada vez que reciba el carácter D (inicial de la palabra decremento), restara una unidad en el contador.

La UART es una componente importante en los subsistema de comunicaciones series, toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, otra UART, toma los bits en bytes completos.

Se denomina “Universal” debido a que el formato de los datos y la velocidad de transmisión son configurables y los niveles de señalización eléctricos y métodos son manejados por un

circuito externo a la UART. Las velocidades de transmisión más comunes son 2400, 4800, 9600, 15500, y 19200 bps. En la Figura 2 se observa la forma que tiene la trama de datos de una UART.

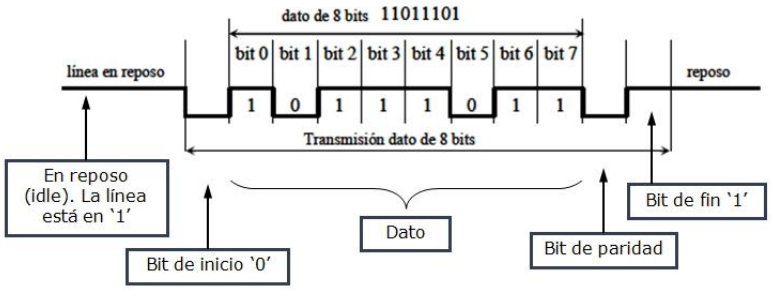

Figura 2. Esquema de transmisión de datos.

El siguiente diagrama de bloque se observa los bloques digitales que constituyen el proyecto.

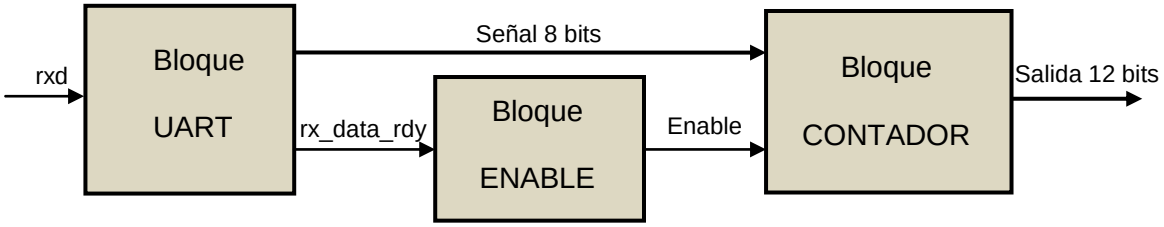

Figura 3. Esquema implementado en el proyecto.

Bloque UART

Este bloque se obtuvo de las practicas de la materia, donde se implementaba un circuito lógico que recibia de una PC, caracteres por medio de una comunicación UART. De acuerdo al carácter recibido, se prendian cuatro led que indicaban, la parte baja o alta del número binario correspondiente al carcater de ingreso (de acuerdo al código Ascii).

En la Figura 4 se observa los archivos que componen este bloque.

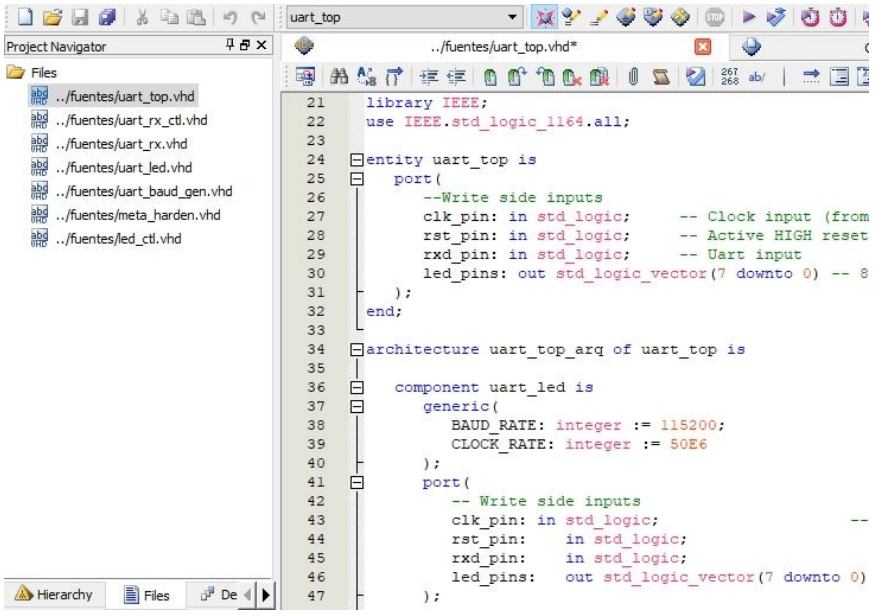

Figura 4. Archivos y código perteneciente al bloque UART.

En este bloque se modificaron los archivos uart_led.vhd y led_ctl.vhd, los cuales incorporaban como entrada la variable btn_pin, que permitia ver los bit altos o bajos en la salida. En este bloque lógico es necesario incorporar en la salida los 8 bits pertenecientes al carácter ingresado, ya que en función de este, el contador incrementara o decrementará la cuenta.En la Figura 5 se observa el diagrama de bloques que se obtuvo con el software Quartus.

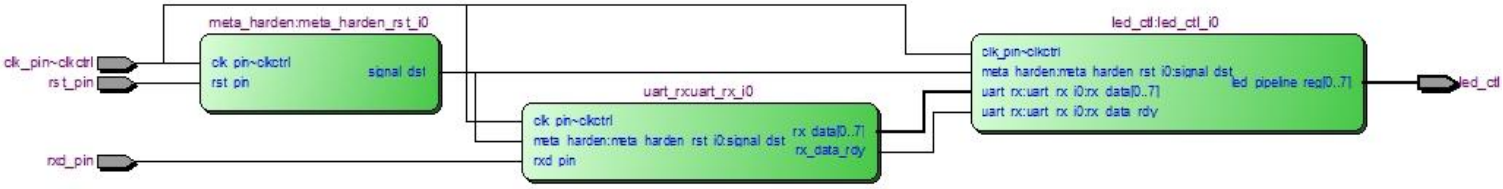

Figura 5. Diagrama de bloque obtenido del software Quartus (Bloque UART).

Para este bloque se realizo la simulación utilizando el software GHDL y GTKwave. En la simulación se observa que se envian tres letras, I, J y K. Esta información ingresa por rxd_pin (pin de entrada) de forma serial. Por ejemplo, para la letra I le corresponde el número binario 01001001. En la salida (led_pins) se observa que sale el 73, de acuerdo a la tabla del código Ascii.

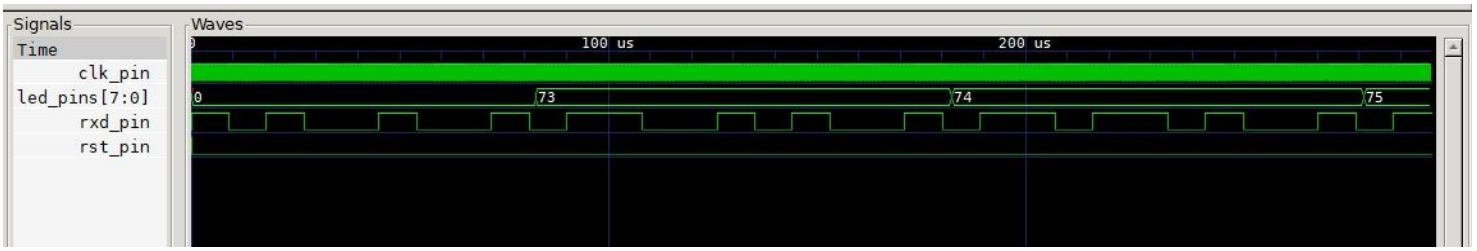

Figura 6. Simulación del Bloque UART.

La simulación concluye al ingresar la letra J(74) y K(75).

Bloque UART-HABILIATACIÓN

A la arquitectura desarrollada previamente, se le agrego un bloque denominado enable.

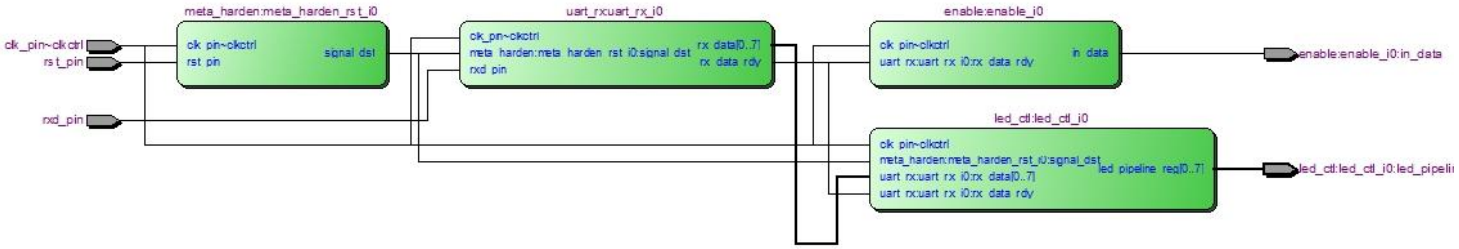

Figura 7. Diagrama de bloque obtenido del software Quartus (Bloque UART-Habilitación).

De acuerdo al esquema desarollado en la Figura 3, la información que ingresa desde la UART pase directamente al contador. Devido a esto, se necesita una señal de habilitación que determinará si el contador realiza un incremento, decremento o ninguna acción en la cuenta.

Del bloque UART se obtiene como salida la variable rx_data_rdy, que inicia en cero y se coloca a uno al tomar la lectura del bit 8 del carácter que ingresa. Esta señal se modifica en el bloque enable y trabaja como señal de habilitación para el contador.

En el siguiente grafico se observa una simulación de este bloque.

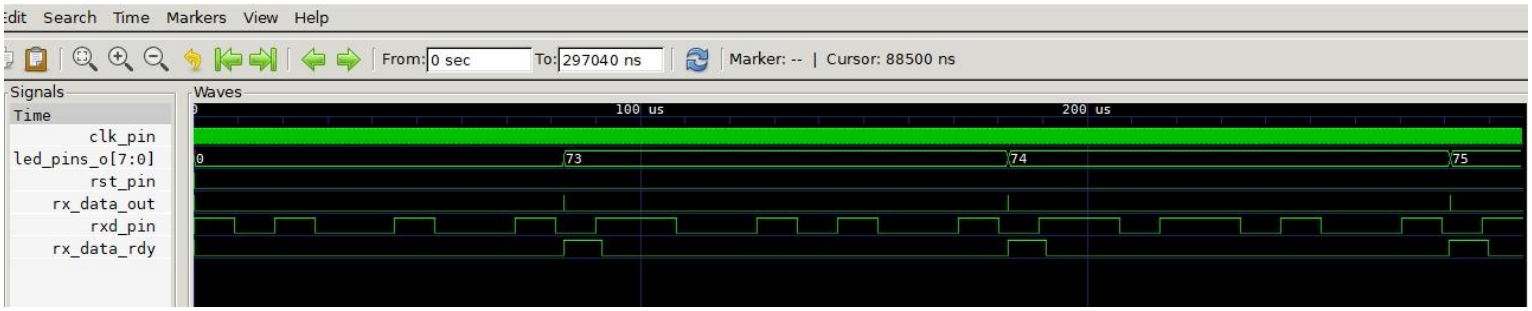

Figura 8. Simulación del Bloque UART-Habilitación.

Se observa el pulso de la variable rx_data_rdy y el pulso denominado rx_data_out, que actua como salida del bloque enable. Esto pulsos se activan cada ves que ingesa una variable por la UART. Para esta simulación ingresaron las letras I, J y K.

Bloque UART-Habilitación-Contador

La estructura desarrollada se conecta al último bloque, el contador.

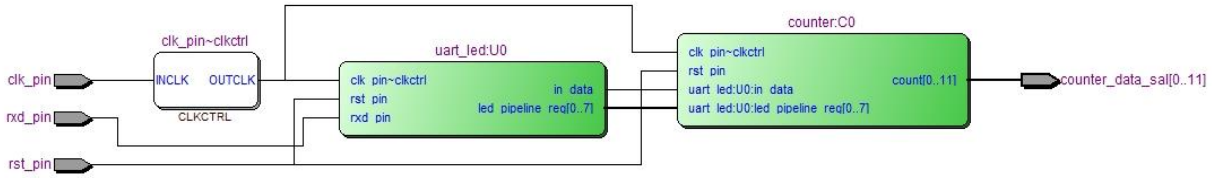

Figura 9. Diagrama de bloque obtenido del software Quartus (Bloque UART-Habilitación).

El contador tiene como entrada la variable reset de la cuenta (ret_pin), el clock (clk), los 8 bits de entrada y el enable, indicado como in_data. La salida del bloque corresponde a 12 señales digitales que se conectarán a un conjunto de resistencias y leds.

En la Figura 10 se observa la simulación de los tres bloques juntos. Se observa la información que ingresa al bloque en forma sereal en la entrada rxd_pin. La información que ingresa al bloque UART pasa al bloque contador, esta información se observa en la salida data_recpt. Al ingresar letra I cuatro veces por rxd_pin, incrementa la cuenta hasta 4 y se mantiene, ya que el siguiente carcater que ingresa es la J. Luego se ingresa la letra I hasta que la cuenta se incremente a 9, para decrecer una unidad por el ingreso de la letra D. Luego crece nuevamente.

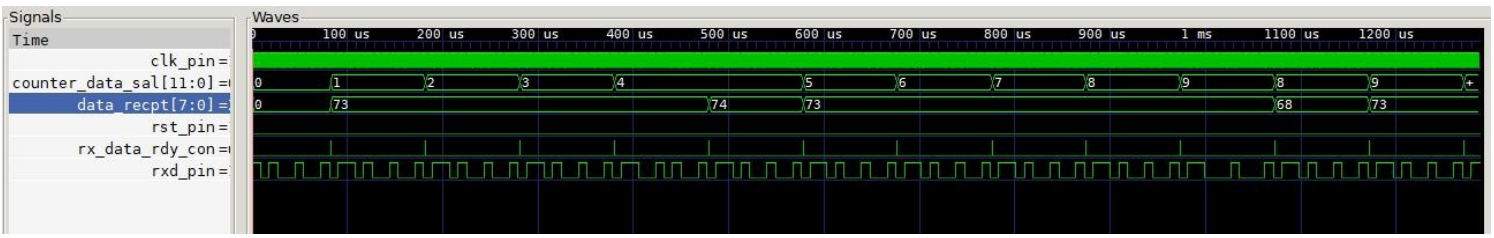

Figura 10. Simulación del Bloque UART-Habilitación-Contador.

Implementación

Se utilizo el software Quartus para depurar el programa y grabar el código. En la siguiente

figura, se observa la selección de los pines para grabar la implementación.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength
clk_pin	Input	PIN_17	1	B1_N0	PIN_17	3.3-V LV...default)		24mA (default)
counter_data_sal[11]	Output	PIN_55	4	B4_N1	PIN_142	3.3-V LV...default)		24mA (default)
counter_data_sal[10]	Output	PIN_53	4	B4_N1	PIN_141	3.3-V LV...default)		24mA (default)
counter_data_sal[9]	Output	PIN_52	4	B4_N1	PIN_7	3.3-V LV...default)		24mA (default)
counter_data_sal[8]	Output	PIN_51	4	B4_N1	PIN_144	3.3-V LV...default)		24mA (default)
counter_data_sal[7]	Output	PIN_48	4	B4_N1	PIN_136	3.3-V LV...default)		24mA (default)
counter_data_sal[6]	Output	PIN_47	4	B4_N1	PIN_3	3.3-V LV...default)		24mA (default)
counter_data_sal[5]	Output	PIN_45	4	B4_N1	PIN_137	3.3-V LV...default)		24mA (default)
counter_data_sal[4]	Output	PIN_44	4	B4_N1	PIN_8	3.3-V LV...default)		24mA (default)
counter_data_sal[3]	Output	PIN_43	4	B4_N1	PIN_143	3.3-V LV...default)		24mA (default)
counter_data_sal[2]	Output	PIN_42	4	B4_N1	PIN_135	3.3-V LV...default)		24mA (default)
counter_data_sal[1]	Output	PIN_41	4	B4_N1	PIN_139	3.3-V LV...default)		24mA (default)
counter_data_sal[0]	Output	PIN_40	4	B4_N1	PIN_9	3.3-V LV...default)		24mA (default)
rst_pin	Input	PIN_63	4	B4_N0	PIN_18	3.3-V LV...default)		24mA (default)
rxd_pin	Input	PIN_28	1	B1_N1	PIN_21	3.3-V LV...default)		24mA (default)

Figura 11. Selección de los pines para la implementación final.

Se selecciona el archivo para grabar el código desarrollado en la FPGA.

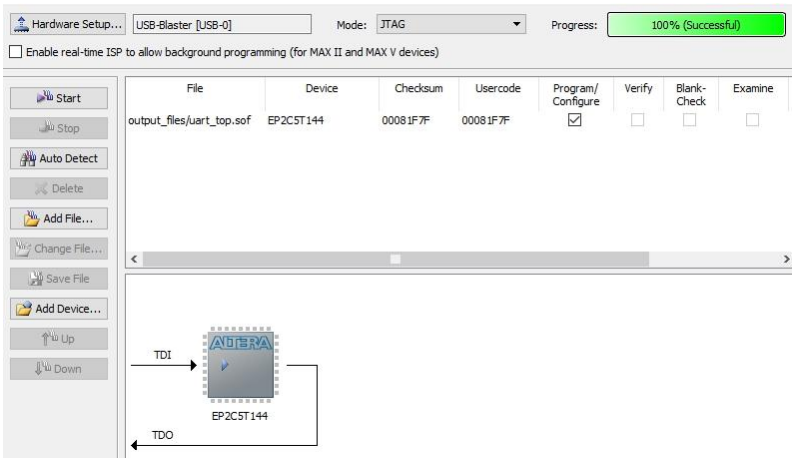

Figura 12. Subiendo el codigo a la FPGA.

Una vez que se grabo el código, se utilizó la interfaz CoolTerm para enviar caracteres desde la PC a la FPGA. Todas las pruebas fueron exitosas.

Conclusiones

Se implementó un contador de 12 bits controlado por UART. Para el sistema digital se utilizó una FPGA Altera Cyclone II EP2C5T144, mediante código VHDL, dos cd4050, un conversor de TTL a UART USB-TTL-PL2303, doce resistencias y doce diodos leds. Se desarrolló el código del contador utilizando bloques lógicos que compartió la cátedra para implementar las prácticas de la materia. Se utilizó el software GHDL y GTKwave para simular y depurar el código del contador. En el trabajo se muestran las simulaciones de cada bloque desarrollado: UART, UART-HABILITACIÓN y UART-HABILITACIÓN-CONTADOR. La implementación se realizó de forma adecuada, la cual se observa en el video donde se observa el funcionamiento del contador.

Anexo

Código para correr el ensayo por medio de GHDL y GTKwave.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity uart_top_tb is
end entity uart_top_tb;
```

```
architecture uart_top_tb_arq of uart_top_tb is

  component uart_top is

                port(
                        --Write side inputs
                        clk_pin: in std_logic;                  -- Clock input (from pin)
                        rst_pin: in std_logic;                  -- Active HIGH reset (from pin)
                        rxd_pin: in std_logic;                  -- Uart input
                        counter_data_sal: out std_logic_vector(11 downto 0)
                );
  end component uart_top;

  constant BAUD_RATE: integer := 115200;
  constant CLOCK_RATE: integer := 50E6;
  signal   clk_tb         : std_logic := '1';
  signal   rst_tb         : std_logic := '1';
  signal   rxd_tb         : std_logic := '1';
  signal   counter_data_sal_tb: std_logic_vector(11 downto 0);
  constant FRECUENCIA    : integer   := 50;              -- en MHz
  constant PERIODO       : time      := 1 us/FRECUENCIA; -- en ns
  signal   detener       : boolean   := false;

begin
  dut: uart_top
              port    map (clk_pin      => clk_tb,
                        rst_pin      => rst_tb,
                                                                        rxd_pin      => rxd_tb,
                        counter_data_sal => counter_data_sal_tb);

  GeneraReloj:
  process begin
    clk_tb <= '1', '0' after PERIODO/2;
    wait for PERIODO;
    if detener then
       wait;
    end if;
  end process GeneraReloj;

  rst_tb <= '1' , '0' after PERIODO*3/2;

  Prueba:
  process is begin
    report "Receptor de la UART a 115200 bits/s y caracteres ASCII"
    severity note;

    wait until rst_tb='0';
    wait for 3 ns;

    rxd_tb <= '1';
    wait for 9 us;

    rxd_tb <= '0';        -- inicio
    wait for 9 us;

    rxd_tb <= '1';        -- bit(0) de la I = "01001001"
    wait for 9 us;
    rxd_tb <= '0';        -- bit(1)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(2)
    wait for 9 us;
    rxd_tb <= '1';        -- bit(3)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(4)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(5)
    wait for 9 us;
    rxd_tb <= '1';        -- bit(6)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(7)
    wait for 9 us;

    rxd_tb <= '1';        -- stop
    wait for 9 us;

    rxd_tb <= '1';        -- idle
    wait for 9 us;

    rxd_tb <= '0';        -- start
    wait for 9 us;

    rxd_tb <= '1';        -- bit(0) de la I = "01001001"
    wait for 9 us;
    rxd_tb <= '0';        -- bit(1)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(2)
    wait for 9 us;
    rxd_tb <= '1';        -- bit(3)
    wait for 9 us;
```

```
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '0';        -- bit(0) de la J = "01001010"
wait for 9 us;
rxd_tb <= '1';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

rxd_tb <= '1';        -- stop
wait for 9 us;
```

```
rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;


        rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
```

```
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

            rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

            rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '0';        -- bit(0) de la D = "01000100"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '1';        -- bit(2)
wait for 9 us;
rxd_tb <= '0';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
rxd_tb <= '0';        -- bit(7)
wait for 9 us;

            rxd_tb <= '1';        -- stop
wait for 9 us;

rxd_tb <= '1';        -- idle
wait for 9 us;

rxd_tb <= '0';        -- start
wait for 9 us;

rxd_tb <= '1';        -- bit(0) de la I = "01001001"
wait for 9 us;
rxd_tb <= '0';        -- bit(1)
wait for 9 us;
rxd_tb <= '0';        -- bit(2)
wait for 9 us;
rxd_tb <= '1';        -- bit(3)
wait for 9 us;
rxd_tb <= '0';        -- bit(4)
wait for 9 us;
rxd_tb <= '0';        -- bit(5)
wait for 9 us;
rxd_tb <= '1';        -- bit(6)
wait for 9 us;
```

```
    rxd_tb <= '0';        -- bit(7)
    wait for 9 us;

                rxd_tb <= '1';        -- stop
    wait for 9 us;

    rxd_tb <= '1';        -- idle
    wait for 9 us;

    rxd_tb <= '0';        -- start
    wait for 9 us;

    rxd_tb <= '1';        -- bit(0) de la I = "01001001"
    wait for 9 us;
    rxd_tb <= '0';        -- bit(1)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(2)
    wait for 9 us;
    rxd_tb <= '1';        -- bit(3)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(4)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(5)
    wait for 9 us;
    rxd_tb <= '1';        -- bit(6)
    wait for 9 us;
    rxd_tb <= '0';        -- bit(7)
    wait for 9 us;

    rxd_tb <= '1';        -- stop
    wait for 9 us;

    detener <= true;
    wait;
  end process Prueba;
end architecture uart_top_tb_arq;
```