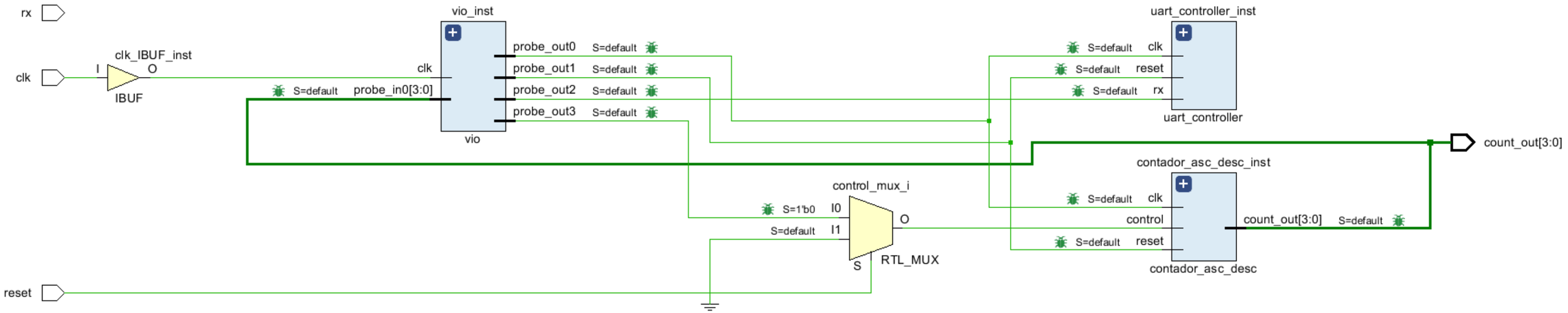
TP FINAL

CONTADOR ASCENDENTE/DESCENDENTE CONTROLADO POR UART

PRESENTADO POR:
ING. JOSÉ ROBERTO CASTRO

SCHEMATIC


rx
clk
clk_IBUF_inst
I
O
IBUF
vio_inst
clk
probe_in0[3:0]
S=default
probe_out0
probe_out1
probe_out2
probe_out3
vio
uart_controller_inst
clk
reset
rx
uart_controller
count_out[3:0]
contador_asc_desc_inst
control
count_out[3:0]
contador_asc_desc
control_mux_i
S=1'b0
I0
I1
O
S
RTL_MUX
reset

RECURSOS

Resource	Utilization	Available	Utilization %
LUT	587	17600	3.34
LUTRAM	24	6000	0.40
FF	1002	35200	2.85
IO	6	100	6.00

ESTRUCTURA

```
DEEPSEEK
  Fuentes
    Arty-Z7-10-Master.xdc
    contador_asc_desc.vhd
    top_level.vhd
    uart_controller.vhd
    uart_rx.vhd
  project_1.cache
  project_1.hw
  project_1.ip_user_files
  project_1.runs
  project_1.sim
  project_1.srcs
  Simulacion
  Sintesis
  project_1.xpr
```

```
##Arty-Z7-10-Master.xdc

## Clock Signal
set_property -dict { PACKAGE_PIN H16    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L13P_T2_MRCC_35 Sch=SYSCLK
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];

## Reset Signal
set_property -dict { PACKAGE_PIN M20    IOSTANDARD LVCMOS33 } [get_ports { reset }]; #IO_L7N_T1_AD2N_35 Sch=SW0

## UART RX Signal
set_property -dict { PACKAGE_PIN M19    IOSTANDARD LVCMOS33 } [get_ports { rx }]; #IO_L7P_T1_AD2P_35 Sch=SW1

## Count Output (LEDs)
set_property -dict { PACKAGE_PIN R14    IOSTANDARD LVCMOS33 } [get_ports { count_out[0] }]; #IO_L6N_T0_VREF_34 Sch=LED0
set_property -dict { PACKAGE_PIN P14    IOSTANDARD LVCMOS33 } [get_ports { count_out[1] }]; #IO_L6P_T0_34 Sch=LED1
set_property -dict { PACKAGE_PIN N16    IOSTANDARD LVCMOS33 } [get_ports { count_out[2] }]; #IO_L21N_T3_DQS_AD14N_35 Sch=LED2
set_property -dict { PACKAGE_PIN M14    IOSTANDARD LVCMOS33 } [get_ports { count_out[3] }]; #IO_L23P_T3_35 Sch=LED3
```

ARTY-Z7-10

FUENTES

```vhdl
-- contador_asc_desc.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity contador_asc_desc is
    Port (
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        control : in STD_LOGIC; -- '0' para ascendente, '1' para descendente
        count_out : out STD_LOGIC_VECTOR(3 downto 0)
    );
end contador_asc_desc;

architecture Behavioral of contador_asc_desc is
    signal count : STD_LOGIC_VECTOR(3 downto 0) := "0000";
begin
    process(clk, reset)
    begin
        if reset = '1' then
            count <= "0000";
        elsif rising_edge(clk) then
            if control = '0' then
                count <= count + 1;
            else
                count <= count - 1;
            end if;
        end if;
    end process;

    count_out <= count;
end Behavioral;
```

```vhdl
-- uart_controller.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity uart_controller is
    Port (
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        rx : in STD_LOGIC;
        control_out : out STD_LOGIC
    );
end uart_controller;

architecture Behavioral of uart_controller is
    signal rx_data : STD_LOGIC_VECTOR(7 downto 0);
    signal rx_done : STD_LOGIC;
begin
    uart_rx_inst : entity work.uart_rx
        port map (
            clk => clk,
            reset => reset,
            rx => rx,
            data_out => rx_data,
            rx_done => rx_done
        );

    process(clk, reset)
    begin
        if reset = '1' then
            control_out <= '0';
        elsif rising_edge(clk) then
            if rx_done = '1' then
                if rx_data = "00000000" then -- Comando para contar ascendente
                    control_out <= '0';
                elsif rx_data = "00000001" then -- Comando para contar descendente
                    control_out <= '1';
                end if;
            end if;
        end if;
    end process;
end Behavioral;
```

FUENTES

```vhdl
-- uart_rx.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity uart_rx is
    Port (
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        rx : in STD_LOGIC;
        data_out : out STD_LOGIC_VECTOR(7 downto 0);
        rx_done : out STD_LOGIC
    );
end uart_rx;

architecture Behavioral of uart_rx is
    signal rx_data : STD_LOGIC_VECTOR(7 downto 0);
    signal rx_done_reg : STD_LOGIC := '0';
begin
    process(clk, reset)
        variable bit_count : integer := 0;
        variable rx_buffer : STD_LOGIC_VECTOR(7 downto 0);
    begin
        if reset = '1' then
            rx_done_reg <= '0';
            bit_count := 0;
        elsif rising_edge(clk) then
            if rx = '0' then -- Start bit detected
                bit_count := 0;
                rx_done_reg <= '0';
            elsif bit_count < 8 then
                rx_buffer(bit_count) := rx;
                bit_count := bit_count + 1;
            elsif bit_count = 8 then
                rx_data <= rx_buffer;
                rx_done_reg <= '1';
                bit_count := 0;
            end if;
        end if;
    end process;

    data_out <= rx_data;
    rx_done <= rx_done_reg;
end Behavioral;
```

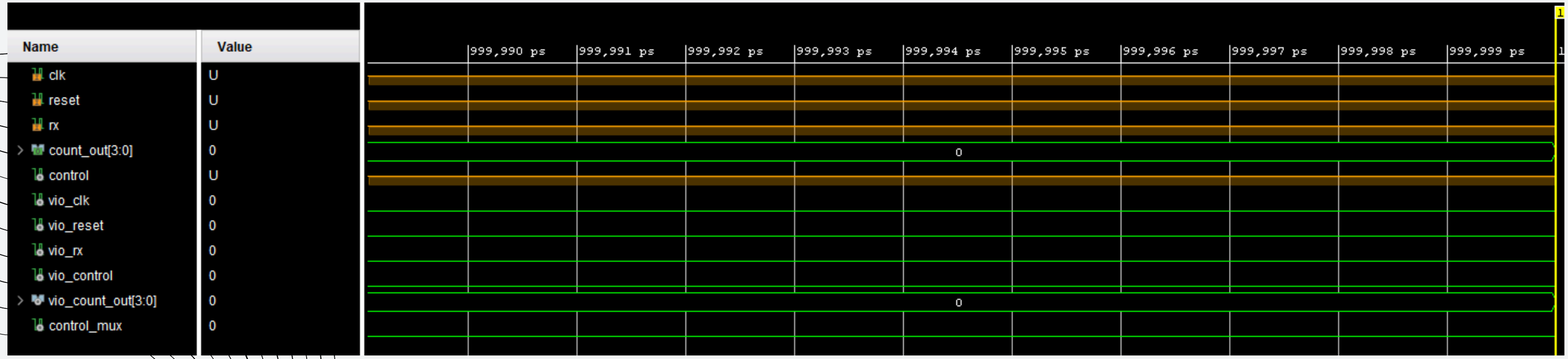
FUENTES

```vhdl
--top_level.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

library work;
use work.all;

entity top_level is
    Port (
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        rx : in STD_LOGIC;
        count_out : out STD_LOGIC_VECTOR(3 downto 0)
    );
end top_level;

architecture Behavioral of top_level is
    signal control : STD_LOGIC;
    signal vio_clk : STD_LOGIC;
    signal vio_reset : STD_LOGIC;
    signal vio_rx : STD_LOGIC;
    signal vio_control : STD_LOGIC;
    signal vio_count_out : STD_LOGIC_VECTOR(3 downto 0);
    signal control_mux : STD_LOGIC;

    COMPONENT vio
        PORT (
            clk : IN STD_LOGIC;
            probe_in0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            probe_out0 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0);
            probe_out1 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0);
            probe_out2 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0);
            probe_out3 : OUT STD_LOGIC_VECTOR(0 DOWNTO 0)
        );
    END COMPONENT;
```

```vhdl
begin
    control_mux <= vio_control when reset = '0' else '0';

    uart_controller_inst : entity work.uart_controller
        port map (
            clk => vio_clk,
            reset => vio_reset,
            rx => vio_rx,
            control_out => control
        );

    contador_asc_desc_inst : entity work.contador_asc_desc
        port map (
            clk => vio_clk,
            reset => vio_reset,
            control => control_mux,
            count_out => vio_count_out
        );

    vio_inst : vio
        PORT MAP (
            clk => clk,
            probe_in0 => vio_count_out, -- 4 bits para count_out
            probe_out0(0) => vio_clk, -- 1 bit para clk
            probe_out1(0) => vio_reset, -- 1 bit para reset
            probe_out2(0) => vio_rx, -- 1 bit para rx
            probe_out3(0) => vio_control -- 1 bit para control
        );

    count_out <= vio_count_out;
end Behavioral;
```

SIMULACIONES

Name	Value	Activity	Direction	VIO
vio_clk	[B] 1		Output	hw_vio_1
vio_control	[B] 0		Output	hw_vio_1
vio_reset	[B] 0		Output	hw_vio_1
vio_rx	[B] 0		Output	hw_vio_1
vio_count_out[3:0]	[H] 0		Input	hw_vio_1
vio_count_out...			Input	hw_vio_1
vio_count_out...			Input	hw_vio_1
vio_count_out...			Input	hw_vio_1
vio_count_out...			Input	hw_vio_1



GRACIAS