Implementación de un contador controlado por UART

Descripción

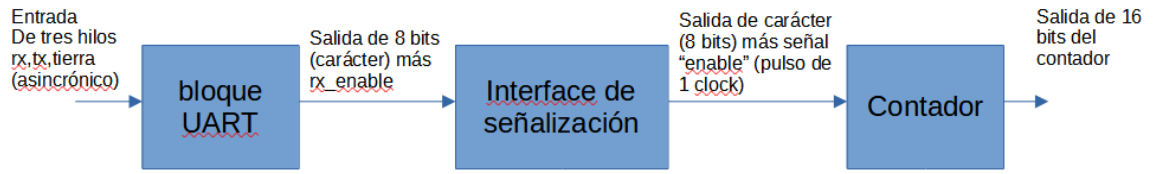
El trabajo consiste en implementar un sistema digital construido usando una FPGA y mediante código VHDL. Básicamente el circuito tendrá como salida un contador de 16 bits sin signo. Cada vez que el sistema reciba por UART un carácter “U” (e.g. *up*) se incrementará en una unidad, mientras que cada vez que se reciba un carácter “D” (e.g. *down*) el contador se decrementará en una unidad.

Se pueden descargar los fuentes desde:

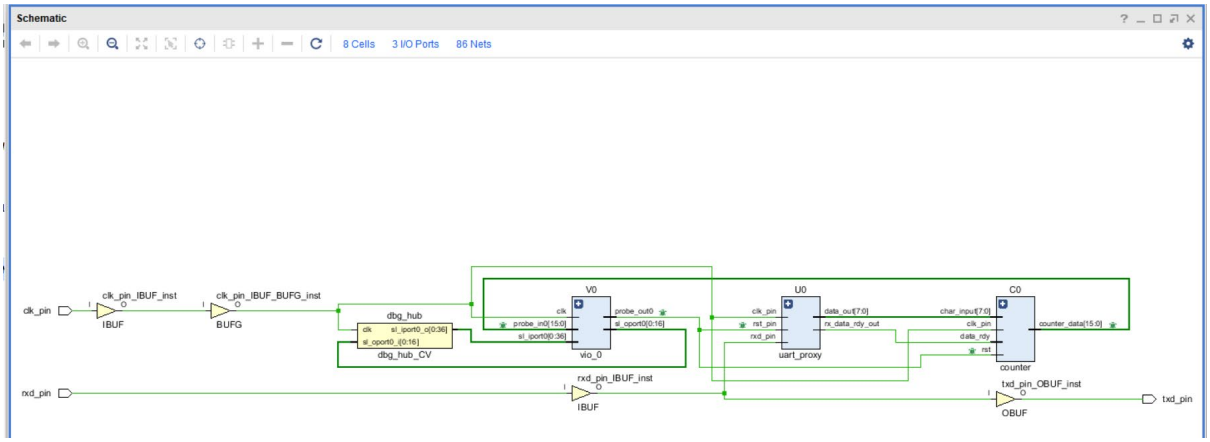
Diagrama en bloques

Partiendo del código provisto por la cátedra de un sistema que tomando una entrada UART, muestra con 4 leds (de la placa de desarrollo Arty Z7-10 de Digilent) los 4 bits la parte menos significativa del carácter recibido. Además, mediante un botón se podía cambiar la salida para mostrar los 4 bits más significativos del carácter recibido. También había una señal de reset (mediante un contacto de entrada) que permitía resetear la salida de leds y apagarlos.

Usando ese proyecto y mediante “ingeniería inversa” del código, se modificó para implementar un sistema que correspondiera al diagrama del flujo de las señales y su procesamiento mostrado abajo.



El diagrama en bloques del sistema implementado:



Simulación

Para la simulación se agregaron dos bloques: un bloque generador de clock de 115200 Hz y un bloque que a partir del clock de temporización enviaba las tramas de prueba. La trama de prueba

consistió en una serie de caracteres precargados. A cada carácter se le agregó el bit de inicio (‘0’) y el bit de stop (‘1’) para enviarlo al puerto del diseño de arriba.

El clock de 115200 Hz sirve para sincronizar los datos (o bits) enviados. Se incia poniendo en ‘0’ en la línea Tx, luego se envían secuencialmente los bits del carácter a enviar y finalmente se coloca de nuevo un ‘1’ en Tx. Además, se mantiene la línea en ‘1’ (bit de stop o línea sin datos) durante tres ciclos al finalizar cada carácter enviado, para asegurar la comunicación.

Cálculo del clock de la UART simulada:

Tiempos (ns)	Referencia
50 ns	Período de clock de FPGA
8,68 us	Período de clock de UART a 115200 bps
173,61	~174 clocks de FPGA = 1 clock de UART ⇨ Cambiar el signo de la línea de salida cada 174/2 = 87 clocks de FPGA para generar el clock UART

Luego se crearon los bloques uart_top_tb.vhd (e.g. test bank) y uart_sim.vhd. Este último bloque simula el puerto serie de la PC. Para realizar la simulación, se envió la trama en formato string: "AUUUlgfsDDDDasd". Se esperaba que el contador subiera hasta 3 y luego bajara hasta “FFFF” (-1 en unsigned).

Luego de muchas idas y vueltas, teniendo que inicializar varios arreglos de señales de componentes internos y también de algunas señales de un bit, se logró realizar la simulación. Los resultados se pueden ver abajo:

La línea data_out[7:0] (con el color más claro) corresponde con el carácter decodificado de la trama enviada: "AUUUlgfsDDDDasd". La última línea es la salida esperada del contador de 16 bits.

Reportes

Temporización (sin errores)

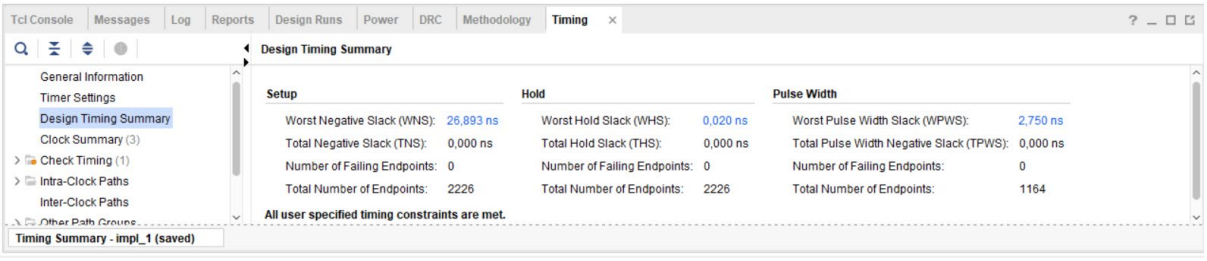
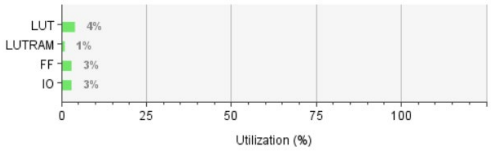
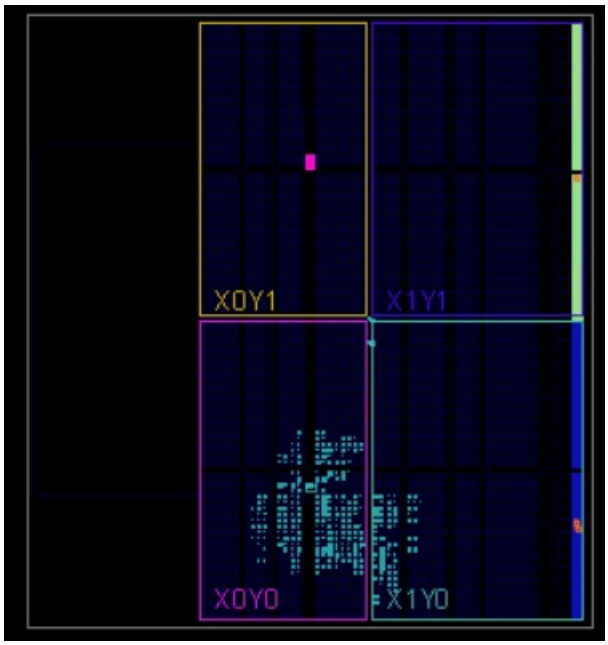


Tabla de utilización

Resource	Utilization	Available	Utilization %
LUT	679	17600	3.86
LUTRAM	24	6000	0.40
FF	1114	35200	3.16
IO	3	100	3.00



Renderizado de la utilización



Implementación

La implementación se realizó en forma remota porque no tengo la placa de desarrollo. Para esto se utilizó tanto el IDE Vivado como una conexión ssh para enviar en forma remota distintos caracteres vía UART. Los resultados obtenidos pueden verse en un video que se encuentra en el repo de Github citado al inicio de este documento.

Partes del código VHDL para destacar

Algunas partes del código que pueden verse abajo, tienen alguna importancia por su impacto en el trabajo. Cada parte tiene una breve descripción y se muestra a que archivo pertenece.

Este bloque tiene como entradas las líneas con el byte decodificado por el bloque de la UART (rx_data) y la señal rx_data_rdy. Cuando se detecta el flanco ascendente de esta señal, se genera un pulso en la salida de 1 clock de duración. Este pulso se envía a la salida junto con el byte recibido para luego contarlo.

```
    process(clk_rx)
    begin
        if rising_edge(clk_rx) then
            if rst_clk_rx = '1' then
                old_rx_data_rdy <= '0';
                char_recpt      <= "00000000";
                new_data_local  <= '0';
            else
                -- Capture the value of rx_data_rdy for edge detection
                old_rx_data_rdy <= rx_data_rdy;
                -- If rising edge of rx_data_rdy, capture rx_data and generate sync pulse
                if (rx_data_rdy = '1' and old_rx_data_rdy = '0') then
                    char_data <= rx_data;
                    new_data_local <= '1';
                else -- else reset pulse
                    new_data_local <= '0';
                end if;
            end if; -- if !rst
        end if;
        char_recpt <= char_data;
        new_data <= new_data_local;
    end process;
```

Fuente: recpt_stm.vhd

Este bloque tiene como entradas el byte recibido por la UART (char_input) y la señal data_rdy de 1 clock de duración. Según el byte de entrada contará 1 valor hacia arriba (letra 'U') o 1 lugar hacia abajo (letra 'D'). Luego mostrará el valor del contador por la salida de 16 bits.

```
     process(clk_pin)
    begin
        if rising_edge(clk_pin) then
            if (rst = '1') then
                count <= 0;
            elsif data_rdy(0) = '1' then -- data_rdy(0) solo dura un pulso
                if char_input = "01010101" then -- char 'U'
                    count <= count+1;
                elsif char_input = "01000100" then -- char 'D'
                    count <= count-1;
                end if;
            end if;
        end if;
    end process;

    counter_data <= std_logic_vector(to_unsigned(count, 16));
```

Fuente: counter.vhd

Este bloque es el principal para poder realizar la simulación.

```
   signal clk_tb: std_logic := '0';
    signal clk_115: std_logic := '0';
    signal tx_rx_aux: std_logic := '1';
begin

    clk_tb <= not clk_tb after 25 ns; -- 20MHz clock
    clk_115 <= not clk_115 after 4341 ns; -- 115200 Hz clock

    UAT: uart_top
        port map(
            rxd_pin => tx_rx_aux,
            clk_pin => clk_tb,
            txd_pin => open
        );

    TX: uart_sim
        port map(
            clk_uart_sim => clk_115,
            uart_tx      => tx_rx_aux
        );
end;
```

Fuente: uart_top_tb.vhd

Este bloque es el encargado de simular la salida UART de una PC.

```
constant msg_to_send: string := "AUUUlgfsDDDDasd";
begin
  process(clk_uart_sim)
      variable starting: integer := 0;
      variable counter: integer := 0;
      variable index: natural := 0;
      variable chr: std_logic_vector(7 downto 0);
      variable tx_aux: std_logic := '1';
  begin
      if rising_edge(clk_uart_sim) then
          if starting < 3 then -- wait for 8,68us x 3 = 26us
              if index < msg_to_send'high then -- does not index reach
the end of string msg?
                  tx_aux := '1';
                  if counter = 0 then -- start bit
                      uart_tx_aux <= '0';
                      index := index + 1;
                      chr :=
std_logic_vector(to_unsigned(character'pos(msg_to_send(index)), 8));
                      tx_aux := '0';
                  elsif counter > 0 and counter < 9 then -- data bits
                      tx_aux := chr(counter - 1);
                  end if;

                  counter := counter + 1;
                  if counter = 10 then  -- counter: bit inicio + 8 char
bits + stop bit = 10 clocks
                      counter := 0;
                  end if;
              end if;
            else
              starting := starting + 1;
            end if;
      end if;
      uart_tx <= tx_aux;
  end process;
```

Fuente: uart_top_tb.vhd