DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Introducción a HDLs (VHDL) - Parte 2

Primer Cuatrimestre 2023

Diseño de Sistemas Digitales con FPGA
DC - UBA

En la clase de hoy

Generación paramétrica 2

Funciones

Atributos

Inferencia

Ejercicios

Generación paramétrica 2

Generación paramétrica

Motivación: implementar un contador decimal de **N** dígitos:
La declaración de esta entidad sería:

```
entity digit_counter is
    generic(B : natural := 10,  -- Base
            N : natural := 4    -- Cantidad de dígitos
            );
    port(clk_i       : in  std_logic;
         reset_i     : in  std_logic;
         run_i       : in  std_logic;
         digit1_o     : out std_logic_vector (natural(ceil(log2(real(B))))-1
                                              downto 0);
         digit2_o     : out std_logic_vector (natural(ceil(log2(real(B))))-1
                                              downto 0);
         -- ????
         max_o       : out std_logic
             );
end entity;
-- ...
```

Generación paramétrica (cont)

Mejor:

```
package digit_counter_pkg is
    constant B: natural :=10;
    constant B_bits : natural := natural(ceil(log2(real(B))));
    type stdlv_array_type is array (integer range <>) of
        std_logic_vector (B_bits-1 downto 0);
end package;
--...
```

La declaración de esta entidad sería:

```
entity digit_counter is
generic(N : natural := 4);     -- Cantidad de dígitos
port(clk_i        : in  std_logic;
     reset_i      : in  std_logic;
     run_i        : in  std_logic;
     -- el rango puede no estar:
     digits_o     : out stdlv_array_type(N-1 downto 0);
     max_o        : out std_logic
     );
end entity;
-- ...
```

Instanciación paramétrica (cont.)

Usemos el `mod_m_counter_prog` de nuevo...

```
-- ...
architecture structural of digit_counter is

    constant max_count : std_logic_vector(B_bits-1 downto 0)
    := std_logic_vector(to_unsigned(B-1, B_bits));

    signal count_next : std_logic_vector(N downto 0);

-- ...

begin

-- ...
```

Instanciación paramétrica (cont.)

Conectando N contadores en cascada ...

```
-- ...
count_next(0) <= run_i;
DIGITS: for i in 0 to N-1 generate
    CONT: entity work.mod_m_counter_prog
         generic map(M => B      -- Modulo
                     )
         port map (clk_i       => clk_i,
                   reset_i     => reset_i,
                   run_i       => count_next(i),
                   max_count_i => max_count,
                   count_o     => digits_o(i),
                   max_o       => count_next(i+1)
                   );
     end generate;
 max_o <= count_next(N);
-- ...
```

Para investigar: `if <cond> generate`

Funciones

Funciones

Toman varios parámetros y devuelven un sólo valor. No son unidades del diseño $\Rightarrow$ **no se pueden sintetizar aisladamente**.

```
package funciones is
    -- ...
    function increment(value : in std_logic_vector; amount : in natural)
        return std_logic_vector;
    -- ...
end package;

package body funciones is
    -- ...
    function increment(value : in std_logic_vector; amount : in natural)
    return std_logic_vector is
        variable result : std_logic_vector(value'range); -- ¿Qué es 'range?
    begin
        result := std_logic_vector(unsigned(value) + to_unsigned(amount, value'
        return result;
    end increment;
    -- ...
end package body;
```

Atributos

Atributos

Proveen información de algún elemento como tips de datos o señales. Se usa la sintaxis `<elemento>'<atributo>`. Por ejemplo para un array `s` cualquiera:

- `s'left, s'right`: los límites izquierdo y derecho del rango de índices de `s`.
- `s'low, s'high`: los límites superiores e inferiores del rango de índices de `s`.
- `s'length`:la longitude del rango de índices de `s`.
- `s'range`: el rango de índices de `s`.
- `s'reverserange`: el rango invertido de índices de `s`.
- etc...

Inferencia

Instanciación de primitivas

Las primitivas se instancian como cualquier componente...
Por ejemplo un Flip-Flop D en Xilinx:

```
-- ...
library unisim; -- Para simulación. Depende del fabricante.
-- ...


    FDRE_INST: FDRE
        port map (
            q => q,
            c => c,
            ce => ce,
            r => r,
            d => d);
-- ...
```

Instanciación de primitivas

Las primitivas se instancian como cualquier componente...
Por ejemplo un Flip-Flop D en Xilinx:

```
-- ...
library unisim; -- Para simulación. Depende del fabricante.
-- ...


    FDRE_INST: FDRE
        port map (
            q => q,
            c => c,
            ce => ce,
            r => r,
            d => d);
-- ...
```

No es código muy portable...

Inferencia

Podemos escribir código de forma que el sintetizador **infiera** lo que queremos...
Por ejemplo una Block-Ram en Xilinx:

```
-- Single-Port RAM with Asynchronous Read (Distributed RAM)

entity rams_dist is
port(clk : in  std_logic;
     we  : in  std_logic;
     a   : in  std_logic_vector(5 downto 0);
     di  : in  std_logic_vector(15 downto 0);
     do  : out std_logic_vector(15 downto 0)
);
end rams_dist;
```

Inferencia (cont.)


```
architecture syn of rams_dist is

type ram_type is array (63 downto 0) of std_logic_vector(15 downto 0);
signal RAM : ram_type;

begin
    process(clk)
    begin
       if (clk'event and clk = '1') then
           if (we = '1') then
                RAM(to_integer(unsigned(a))) <= di;
           end if;
       end if;
    end process;

    do <= RAM(to_integer(unsigned(a)));
end syn;
```

Inferencia (cont.)


```
architecture syn of rams_dist is

type ram_type is array (63 downto 0) of std_logic_vector(15 downto 0);
signal RAM : ram_type;

begin
    process(clk)
    begin
       if (clk'event and clk = '1') then
           if (we = '1') then
                RAM(to_integer(unsigned(a))) <= di;
           end if;
       end if;
    end process;

    do <= RAM(to_integer(unsigned(a)));
end syn;
```

¡¡¡Hay que leer la documentación!!!

Synthesis docs

- **ISE:** `https://www.xilinx.com/htmldocs/xilinx11/xst.pdf`
- **Vivado:** `https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2022_2/ug901-vivado-synthesis.pdf`

Ejercicios

stopwatch

Planteamos en este ejercicio realizar un reloj con 3 dígitos decimales y cuenta de 00.0 a 99.9 segundos y cuando llega al máximo vuelve a empezar. Contiene una señal sincrónica de `clr` que regresa la cuenta a 00.0 y una señal de enable, llamada `en`, que habilita o suspende la cuenta. El diseño es básicamente un contador BCD [1]. En este formato, un número decimal es representado por 4 dígitos binarios. Por ejemplo el número 452_{10} es representado como "0100 0101 0011".

[1] `https://www.wikiwand.com/es/Decimal_codificado_en_binario`

Notas de implementación (Nexys 2)

- Tengan en cuenta que disponen de un cristal de 50 MHz en la placa y van a tener que utilizar contadores para obtener el período más corto (0.1 s)
- El acceso a los displays 7 segmentos se realiza de manera multiplexada, para ahorrar pines, como está explicado en "Outputs: Seven-Segment Display"[2], de manera que tienen que ir iluminando los displays uno a uno, según una tasa de refresco determinada (recomendamos entre 60 Hz y 1 kHz). Son 4 displays y cada uno cuenta con una señal de enable y sus 8 leds. Esto equivale a que tenemos 12 bits (8 bits de leds y 4 de enable) para acceder a los 4 displays. Todas las señales son activo bajo.

[2] `https://digilent.com/reference/programmable-logic/nexys-2/reference-manual`

stopwatch++ (optativo)

- Agreguen a su stopwatch una señal `up` que cuente en forma ascendente si está alta y descendente en caso contrario.
- Agreguen un indicador de minutos de 0 a 9, de manera que quede M.SS.D.