

Diseño Digital I

Modelos VHDL parametrizables

Modelado con VHDL de un Banco de registros

BANCO DE 32 REGISTROS DE 32 BITS

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

entity reg_file_32_32bits is
port(clk:      in      std_logic;
     rst_n:    in      std_logic;
     WR:       in      std_logic;
     Dir_WR:   in      std_logic_vector(4 downto 0);
     Dir_RD:   in      std_logic_vector(4 downto 0);
     Dato_in:  in      std_logic_vector(31 downto 0);
     Dato_out: buffer std_logic_vector(31 downto 0));

end entity;

architecture rtl of reg_file_32_32bits is
    type reg_file_32bits is array ( NATURAL RANGE <>) of std_logic_vector(31 downto 0);
    signal reg_file_op: reg_file_32bits(31 downto 0);

begin
    process(clk, rst_n)
    begin
        if rst_n = '0' then
            for i in 31 downto 0 loop
                reg_file_op(i) <= (others => '0');
            end loop;

            elsif clk'event and clk = '1' then
                if WR = '1' then
                    reg_file_op(conv_integer(Dir_WR)) <= Dato_in;

                    end if;
                end if;
            end process;

            Dato_out <= reg_file_op(conv_integer(Dir_RD));

        end rtl;
```



Modelado con VHDL de un Banco de registros parametrizable (I)

- Puede resultar conveniente disponer de modelos de sistemas en los que se puedan parametrizar algunas características
- Por ejemplo, un banco de registros en el que se pueda elegir el número de registros del banco y la longitud en bits de cada uno

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

use work.auxiliar.all;

entity reg_file_M_Nbits is

generic(N: in natural := 32;    -- Número de bits
        M: in natural := 32);  -- Número de registros

port(rst_n:    in  std_logic;
     clk:      in  std_logic;
     WR:       in  std_logic;
     Dir_WR:   in  std_logic_vector(ceil_log(M)-1 downto 0);
     Dir_RD:   in  std_logic_vector(ceil_log(M)-1 downto 0);
     Dato_in:  in  std_logic_vector(N-1 downto 0);
     Dato_out: out std_logic_vector(N-1 downto 0));
end entity;
```

PAQUETE DE USUARIO

Paquete que contiene una función (**ceil_log**) que calcula **ceil(log(n))**

VALORES QUE PARAMETRIZAN EL MODELO (GENÉRICOS)

Son constantes cuyo valor define las dimensiones (número de registros y de bits de cada uno) del banco

PUERTOS PARAMETRIZADOS

Definición de una función en un Paquete VHDL

```
package auxiliar is
  -- La función ceil_log calcula el menor número
  -- natural, n, para el que 2**n es >= que x
  function ceil_log(x: in natural) return natural;
end package;
```

DECLARACIÓN DEL PAQUETE Y
DE LA FUNCIÓN

```
package body auxiliar is
  function ceil_log(x: in natural) return natural is
  begin
    for n in 1 to 32 loop
      if 2**n >= x then
        return n;

      end if;
    end loop;
    return 0; --Error
  end ceil_log;
end package body;
```

CUERPO DEL PAQUETE Y DE LA
FUNCIÓN

Modelado con VHDL de un Banco de registros parametrizable (II)

```
architecture rtl of reg_file_M_Nbits is
  -- M: número de registros
  -- N: número de bits de cada registro
  type reg_file is array (NATURAL RANGE <>) of std_logic_vector(N-1 downto 0);
  signal reg_file_op: reg_file(M-1 downto 0);

begin

  process(clk, rst_n)
  begin
    if not rst_n then
      for i in reg_file_op'range loop
        reg_file_op(i) <= (others => '0');
      end loop;

    elsif clk'event and clk = '1' then
      if WR then
        if conv_integer(Dir_WR) < M then
          reg_file_op(conv_integer(Dir_WR)) <= Dato_in;
        end if;
      end if;
    end if;
  end process;

  Dato_out <= reg_file_op(conv_integer(Dir_RD)) when conv_integer(Dir_RD) < M
    else (others => 'X');

end rtl;
```

USO DE PARAMETROS
EN EL MODELADO DEL
FUNCIONAMIENTO

ATRIBUTO VHDL

reg_file_op'range va
desde M-1 hasta cero

USO DE PARAMETROS
EN EL MODELADO DEL
FUNCIONAMIENTO

USO DE PARAMETROS
EN EL MODELADO DEL
FUNCIONAMIENTO

Atributos

- Los **atributos** VHDL proporcionan información adicional sobre el elemento del lenguaje sobre el cual son aplicados: un tipo de datos, un rango, un valor (de un determinado tipo).
- Pueden aplicarse distintos elementos del lenguaje: tipos de datos, objetos, etc.
- Pueden ser de diferentes tipos: valores, tipos, rangos, funciones o señales.
- Ejemplos:
 - **std_logic'left** es '**U**': '**left** es un atributo de tipo de datos; es un valor del mismo tipo que el dato al que se aplica (el valor "izquierdo" del tipo de datos).
 - **S'stable(T)** es una señal booleana cuyo valor es **TRUE** si **S** no ha cambiado de valor durante el periodo de tiempo **T**. '**stable** es un atributo de tipo señal (booleana) que se aplica a señales de cualquier tipo de datos. **No puede emplearse en modelos RTL.**

Modelo parametrizable de un decodificador

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

entity decod_N_a_exp2N is
generic(N: in natural := 5); -- Bits del dato de entrada

port(ena:      in  std_logic;
     Dato_in:  in  std_logic_vector(N-1 downto 0);
     Dato_out: out std_logic_vector(2**N -1 downto 0));
end entity;

architecture rtl of decod_N_a_exp2N is
begin

process(all)
begin
    for i in Dato_out'range loop
        if i = conv_integer(Dato_in) then
            Dato_out(i) <= ena;

        else
            Dato_out(i) <= '0';
        end if;
    end loop;
end process;

end rtl;
```

**USO DE PARAMETROS
EN EL MODELADO DE
LA INTERFAZ**

**USO DE PARAMETROS
EN EL MODELADO DEL
FUNCIONAMIENTO**

Modelo parametrizable de un registro

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

entity reg_Nbits is
generic(N: in integer := 32);

port(rst_n:      in  std_logic;
     clk:        in  std_logic;
     ena:        in  std_logic;
     Dato_in:    in  std_logic_vector(N-1 downto 0);
     Dato_out:   out std_logic_vector(N-1 downto 0));
end entity;

architecture rtl of reg_Nbits is
begin

    process(clk, rst_n)
    begin
        if not rst_n then
            Dato_out <= (others => '0');

        elsif clk'event and clk = '1' then
            if ena then
                Dato_out <= Dato_in;
            end if;
        end if;
    end process;
end rtl;
```

**USO DE PARAMETROS
EN EL MODELADO DE
LA INTERFAZ**

**MODELADO DEL
FUNCIONAMIENTO
INDEPENDIENTE DEL
PARÁMETRO**

Modelado estructural parametrizable

```
architecture str of reg_file_M_Nbits is
  -- M: número de registros
  -- N: número de bits de cada registro
  type reg_file is array (NATURAL RANGE <>) of std_logic_vector(N-1 downto 0);
  signal reg_file_op: reg_file(M-1 downto 0);
  signal WR_reg:      std_logic_vector(M-1 downto 0);
}

begin
  G1: for i in 0 to M-1 generate
    reg: entity work.reg_Nbits(rtl)
      generic map(N => N)
      port map(clk      => clk,
               rst_n    => rst_n,
               ena      => WR_reg(i),
               Dato_in   => Dato_in,
               Dato_out  => reg_file_op(i));
  end generate;

  decod: entity work.decod_N_a_exp2N(rtl)
    generic map(N => ceil_log(M))
    port map(ena      => WR,
             Dato_in   => Dir_WR,
             Dato_out  => WR_reg);

  Dato_out <= reg_file_op(conv_integer(Dir_RD)) when conv_integer(Dir_RD) < M
    else (others => 'X');

end str;
```

**NODOS DE CONEXIÓN
PARAMETRIZADOS**

SENTENCIA **generate**

Genera **M** sentencias de
emplazamiento

**USO DE PARAMETROS
EN LAS SENTENCIAS DE
EMPLAZAMIENTO**

**SENTENCIA
CONCURRENTE EN
MODELO
ESTRUCTURAL**

Sentencia Generate

- Las sentencias **generate** permiten generar estructuras regulares
 - Réplicas de partes del código

ETIQUETA: **FOR** (ITERACIÓN) **GENERATE**
SENTENCIAS CONCURRENTES

END GENERATE;

ETIQUETA: **IF** (CONDICIÓN) **GENERATE**
SENTENCIAS CONCURRENTES

END GENERATE;

```
g0: for i in 0 to N-1 generate
    S(i) <= E(i) and ctrl;
end generate;
```



POLITÉCNICA



ETSIT
UPM

escuela técnica superior de
ingeniería
de
diseño
industrial



Telecomunicación
Campus Sur
UPM

MICROELECTRÓNICA
μe-UPM



INSTITUTO
DE ENERGÍA
SOLAR



CEI UPM | Centro de
Electrónica
Industrial

CENTRO
LÁSER
UPM
UNIVERSIDAD
POLITÉCNICA
DE MADRID

ISOM



citSem

➤ Más información: <https://blogs.upm.es/ue-upm/>

➤ Contacto: comunidad.microelectronica@upm.es



MINISTERIO
DE CIENCIA, INNOVACIÓN
Y UNIVERSIDADES



Financiado por
la Unión Europea
NextGenerationEU



Plan de Recuperación,
Transformación y
Resiliencia



AGENCIA
ESTATAL DE
INVESTIGACIÓN

UNIÓN EUROPEA
Fondos estructurales
Invertimos en su futuro



UNIÓN EUROPEA
Fondo Social Europeo
El Fondo Social Europeo invierte en tu futuro



**Comunidad
de Madrid**

PERTE Chip
microelectrónica y
semiconductores



GOBIERNO
DE ESPAÑA

MINISTERIO
PARA LA TRANSFORMACIÓN DIGITAL
Y DE LA FUNCIÓN PÚBLICA

MICROELECTRÓNICA
μe-UPM