Cómo utilizar un package en un proyecto de VHDL

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/05/18/como-utilizar-un-package-en-un-proyecto-de-vhdl/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

Si alguna vez has tenido un proyecto de gran envergadura, te habrás dado cuenta que cuanto más grande, más complejo de manejar todo. Por ello lo mejor que puedes haces es hacerlo lo más genérico que puedas, utilizando constantes y genéricos en todos los sitios que sea posible, además de otras estructuras con *others*.

Bien pues una forma de manejar todo eso de forma más fácil es empaquetarlo todo en un *package*. La ventaja es que solo modificando un parámetro modificas todos los ficheros, tamaños de variables, etc.

Para ello, lo primero, debido a que se suelen trabajar con formatos std_logic es declarar la librería de std_logic.

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

Lo siguiente es definir un *package*. Si revisas la teoría de los *package* en internet verás que los *package* tienen dos partes, una en la que se definen constantes y las funciones y otra en la que se implementan las funciones(*function*) y los procedimientos(*procedure*).

Bien, en nuestro caso, solo utilizaremos la primera forma. Para ello definimos el *package* con el nombre. (el fichero se tiene que llamar igual que el *package*)

```
package <nombre> is
...
end package;
```

Es importante que el nombre del *package* sea definitorio de lo que contiene y que sea fácilmente reconocible, para ello se recomienda el uso de mayúsculas y que se llame, «CONSTANTES», «VALORES», etc

Para las definiciones primero se recomienda definir las constantes que vayan primero a los genéricos del fichero ***top***, y de estos vayan cayendo fichero a fichero, para que al modificar una constante se modifiquen todos los ficheros.

También es interesante que queden reflejadas estas constantes antes de crear los *std_logic_vector*, porque aquí aparece una estructura interesante que nos evita tener que usar los *std_logic_vector* en cada definición que es la instrucción *subtype* de VHDL, que permite renombrar un tipo. Se definiría de la siguiente forma.

```
subtype <nombre del tipo> is std_logic_vector(<genérico>-1 downto 0);
```

¿Para qué hacer eso? Para que al definir las constantes se utilice el *subtype* creado. **NOTA**: **es importante que al crear una constante, esta quede bien clara. Para ello se recomienda el uso de mayúsculas y un prefijo del tipo «PK_» para que sea fácilmente reconocible dónde buscar el valor, tipo «PK_ADD», o «PK_Cinco», algo que sea reconocible.**

```
constant <constante nombre>       : <subtype creado> := <valor constante>;
```

Esto nos facilita la vida para evitar tener que modificar todos los ficheros de VHDL al modificar un parámetro.

Pongo un ejemplo de *package* genérico.

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

package PK_VALUES is
    constant PK_ALU_WIDTH : integer := 10;


-- Alu constants
    subtype alu_type is std_logic_vector(PK_ALU_WIDTH-1 downto 0);
    constant PK_ADD_select     : alu_type := (0=>'1',others=>'0');
    constant PK_SUB_select     : alu_type := (1=>'1',others=>'0');
    constant PK_SLL_select     : alu_type := (2=>'1',others=>'0');
    constant PK_SLT_select     : alu_type := (3=>'1',others=>'0');
    constant PK_SLTU_select    : alu_type := (4=>'1',others=>'0');
    constant PK_XOR_select     : alu_type := (5=>'1',others=>'0');
    constant PK_SRL_select     : alu_type := (6=>'1',others=>'0');
    constant PK_SRA_select     : alu_type := (7=>'1',others=>'0');
    constant PK_OR_select      : alu_type := (8=>'1',others=>'0');
    constant PK_AND_select     : alu_type := (9=>'1',others=>'0');

end package;
```

Para poder usar estas constantes en un fichero se tienen que definir como una librería tipo *work*.

```
library work;
use work.<nombre package>.all;
```

En el ejemplo anterior.

```
...
library work;
use work.PK_VALUES.all;
...
generic map(
     ALU_WIDTH => PK_ALU_WIDTH,
...
when state is
    case PK_ADD_select => ...
    case PK_SUB_select => ...
```

NOTA

Vivado tiene el inconveniente de que los package no los muestra con el resto de ficheros, sino que para poder verlo tienes que ir la pestaña «*Compile Order*» de «*Sources*«.