Estándar VHDL del 2008

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/11/26/estandar-vhdl-del-2008/

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

GitHub: https://github.com/DRubioG

Fecha última modificación: 23/02/2025

En esta entrada explicaré alguna de las nuevas cosas que incluye el estándar del 2008. El estándar del 2008 es muy amplio, tiene muchas cosas nuevas y algunas poco útiles en el desarrollo de firmware avanzado.

***NOTA**: yo personalmente **NO** recomiendo este estándar, es mucho ruido y pocas nueces. Además, trata de convertir el lenguaje de descripción de hardware, VHDL, en un lenguaje de programación estándar. Cosa que no es. Además, es un estándar conflictivo con los otros estándares e incorpora estructuras no deterministas en el lenguaje que pueden no ser compatibles con los estándares de certificación.*
Aclaración, **el problema no es cómo utilice el nuevo estándar una persona con experiencia en VHDL, el problema es cómo lo puede llegar a utilizar una persona sin experiencia en VHDL y el riesgo que eso puede representar. *En diseño firmware las «ideas felices» con un conocimiento pobre del lenguaje y de sus estructuras estádares de desarrollo, se pagan bien caro (y Dios no quiera que tengas que ser tú el bombero que apague ese fuego, y lo digo con conocimiento de causa, de dolores y de muchas cosas más).***

process(all)

Este nuevo tipo de process permite incluir la palabra reservada *all* en la lista de sensibilidad, por lo que delega la responsabilidad de localizar las señales asíncronas en el sintetizar, de tal manera que el usuario no se tiene que preocupar por localizar o conocer que posibles señales son asíncronas.

```
process(all)
begin
   if rst_n = '0' then
      c <= '1';
   elsif rising_edge(clk) then
       c <= '1';
       if a  = '1' then
           c <= '0';
       end if;
   end if;
end process;
```

/* Comentarios */

Desde esta nueva versión se pueden utilizar los comentarios multilínea de otros lenguajes de programación.

```
/*
   Esto es un comentario multilínea en VHDL 2008
*/
```

elsif … generate

Esta es una de las cosas más interesantes de VHDL 2008, en esta versión se permiten crear *generates* en cadena, cosa que anteriormente no se podía.

```
inst_generate : if a = "alfa" generate
   inst_alfa : alfa
      port map ( ... );
elsif a = "beta" generate
   inst_beta : beta
       port map( ... );
end generate;
```

case … generate

Al igual que en el estándar se puede usar el *elsif*, ahora también se puede utilizar el *case … generate*.

```
inst_generate : case a generate
    when "alfa" =>
         inst_alfa : alfa
             port map ( ... );
    when "beta" =>
         inst_beta : beta
             port map( ... );
end generate;
```

case?

Ahora los *case* no hace falta que estén fuertemente definidos, solo hace falta definir lo mínimo que quieras detectar.

```
case? a is
    when "-1" => b <= '1';
    when "1-" => b <= '0';
    when others => null;
end case?;
```

Operaciones en instanciaciones

Esta es otra cosa interesante que añade VHDL 2008. Ahora permite hacer operaciones en la instanciación de módulos.

```
inst_bloque : bloque
   port map(
       a => not j,
       b => "00" & r
   );
```

Nuevo sistema de bases

VHDL solo permite un número determinado de bases numérica, como el binario, el hexadecimal o el octal. El resto de representaciones o se basan en binario o son hacen con *others* o se hace con una conversión de un integer. Bien pues ahora VHDL permite crear una nueva base especifica para el dato.

```
constant seis_bits : std_logic_vector(5 downto 0) := 6x"01";
```

Operaciones de un bit de salida

Ahora VHDL permite hacer operaciones binarias usando funciones por defecto incluidas. Operaciones como: *and, or, nand, xor, nor y xnor.* Anteriormente te tenías que valor de la librería *std_logic_misc*, esto último ya lo he contado en esta entrada anterior.

https://soceame.wordpress.com/2024/05/18/como-hacer-una-and-or-xor-nand-etc-de-todos-los-bits-de-una-senal-en-vhdl/

```
salida <= nand("001010");
```

?=

Ahora se pueden hacer comparaciones en una línea.

```
--antes
if c = b then
    a <= '1';
else
    a <= '0';
end if;
-- VHDL'08
a <= b ?= c;
```

Condicionales binarias

Ahora no hace falta definir las condicionales binarias.

```
if a then
   b <= '1';
else
   b <= '0';
end if;
```

Lectura de puertos de salida

Ahora se pueden leer los puertos de salida sin definirlos como *inout*

```
port( ...
    a : out std_logic_vector(...);
    b : out std_logic;
```

```
);
...
if a = x"55" then
    b <= '1';
else
    b <= '0';
end if;
```

El resto de estructuras del estándar son estructuras con más ruido que utilidad.