Cómo mejorar tus simulaciones en VHDL

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/05/26/como-mejorar-tus-simulaciones-en-vhdl/

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

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

Hay dos trucos que permiten optimizar y mejorar las simulaciones en VHDL.

- Utilizar un report
- Utilizar procedimientos

Report

report es una sentencia que se puede utilizar para imprimir un mensaje cuando una simulación llega a un punto, se puede utilizar para marcar un cambio en la simulación.

```
report "<mensaje>" [severity [note/warning/error/failure]];
```

Bien, pues el ***severity failure*** para la simulación en seco, por lo que lo puedes utilizar para ver que es *lo que ocurre cuando la simulación llega a ese punto.* Además, dependiendo de como hayas hecho el testbench puedes continuar simulando después de la parada.

```
report "para" severity failure;
```

Procedure

Los procedure son estructuras que permiten encapsular el código y facilitar la repetición de este. Además no devuelven un valor como las funciones, sino que modifican las señales que se le da. Siguen la siguiente estructura.

```
procedure <nombre> ( <puertos> ) is
begin

end procedure;
```

Lo vemos en un ejemplo:

Yo quiero que la señal ‘a’ cambie cada 20ns a un valor del siguiente orden: x»01″, x»07″, x»10″, x»FF», x»00″, x»04″, x»0C» y x»AA» pero cada 10ns vuelva a x»00″. Para ello de primeras tenemos que definir explícitamente la ejecución.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity leds_tb is
end leds_tb;

architecture Behavioral of leds_tb is
...
begin
...
 process begin
   a <= x"00";
   wait for 10ns;
   a <= x"01";
   wait for 10ns;
```

```
   a <= x"00";
   wait for 10ns;
   a <= x"07";
   wait for 10ns;
   a <= x"00";
   wait for 10ns;
   a <= x"10";
   wait for 10ns;
   a <= x"00";
   wait for 10ns;
   a <= x"FF";
   wait for 10ns;
   a <= x"00";
   wait for 10ns;
   a <= x"00";
   wait for 10ns;
   a <= x"00";
   wait for 10ns;
   a <= x"04";
   wait for 10ns;
   a <= x"00";
   wait for 10ns;
   a <= x"0C";
   wait for 10ns;
   a <= x"00";
   wait for 10ns;
   a <= x"AA";
 end process;

end Behavioral;
```

Como se puede ver esto es un coñazo, pues imagina sistemas más grandes, como simular una cadena de UART a nivel binario.

Para ello están los procedimientos, que permiten pasar de lo anterior a lo siguiente.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity leds_tb is
end leds_tb;

architecture Behavioral of leds_tb is
procedure envio(
              input : out std_logic_vector(7 downto 0);
              value : std_logic_vector(7 downto 0)) is
begin
   input <= x"00";
   wait for 10ns;
   input <= value;
   wait for 10ns;
end procedure;
...
begin
...
 process begin
   envio(a, x"01");
   envio(a, x"07");
```

```
   envio(a, x"10");
   envio(a, x"FF");
   envio(a, x"00");
   envio(a, x"0C");
   envio(a, x"AA");
   report "para" severity failure;
 end process;

end Behavioral;
```

Entonces, como parámetros del *procedure* se mete la señal ‘a’, que viene de un puerto de entrada, y sale del *procedure* con el valor que se le quiera asignar. El valor «value» no tiene tipo porque es una constante.

Además, se puede utilizar un *procedure* de otro *procedure*.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity leds_tb is
end leds_tb;

architecture Behavioral of leds_tb is

procedure envio(
               input : out std_logic_vector(7 downto 0);
               value : std_logic_vector(7 downto 0)) is
begin
   input <= x"00";
   wait for 10ns;
   input <= value;
   wait for 10ns;
end procedure;

procedure proyecto(input : out std_logic_vector(7 downto 0)) is
begin
   envio(input, x"01");
   envio(input, x"07");
   envio(input, x"10");
   envio(input, x"FF");
   envio(input, x"00");
   envio(input, x"0C");
   envio(input, x"AA");
   report "para" severity failure;
end procedure;
...
begin
...
 process begin
   proyecto(a);
 end process;

end Behavioral;
```

Pues la gran ventaja de esta forma de encapsular el código del testbench es que se puede meter en packages, para que no molesten en el código a simular, así se pueden manejar mejor.

NOTA: Por último, también recuerdo que la directiva «wait;» se puede declarar sola y evita la repetición de código dentro de un «process begin»