Cómo generar números aleatorios en una simulación de VHDL

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2025/02/16/como-generar-numeros-aleatorios-en-una-simulacion-de-vhdl/

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

GitHub: https://github.com/DRubioG

Fecha última modificación: 16/02/2025

Todo lo que comentará ahora es sólo para simulación. Debido a que la librería math_real solo funciona en simulación, por lo que no es sintetizable.

Para generar números aleatorios en VHDL existe un procedure dentro de la librería math_real llamado uniform que es el que se utiliza para generar números aleatorios en VHDL.

Este procedure se define de la siguiente forma.

- `procedure UNIFORM (variable Seed1,Seed2:inout integer; variable X:out real);`

Donde seed1 y seed2 son variables de tipo integer. Y rand de tipo real. Esto es importante debido a que la salida de esta función es un valor decimal entre 0 y 1. Por lo que si se quiere conseguir un valor de salida que esté entre 0 y 100 se tiene que multiplicar por 100 y además, es necesario cambiar el tipo de valor a integer para que se convierta en un número entero.

NOTA: los valores de la variable seed van de 1 a 2147483398.

Es importante también entender que seed1 y seed2 son de tipo variable, por lo que este procedure solo puede estar contenido en algo que admita variables como un *process*, un *package* o un *function*.

Ejemplo

Imaginemos que queremos generar números aleatorios entre 0 y 100 en VHDL para una simulación cada 10ns. Bien para ello lo primero es definir la librería.

```
use ieee.math_real.all;
```

Lo siguiente que hacemos es definir el *process* donde lo vamos a utilizar.

```
process
   variable seed1: integer:=0 ;
   variable seed2: integer:=99;
   variable rand : real;
begin
    uniform(seed1, seed2, rand);
    b <= rand;
    a <= integer(rand*100.0);
    wait for 10ns;
end process;
```

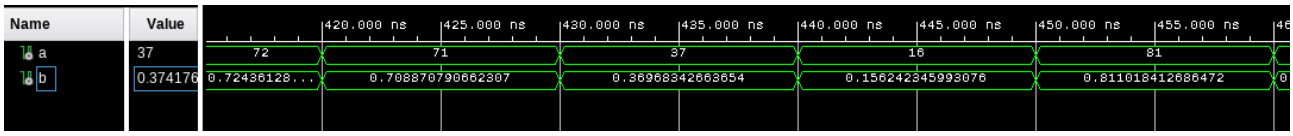
En el *process* anterior se puede ver que ponemos dos valores como “semillas”(seed1/2) estos valores son para hacer la aleatoriedad, y son necesarios, no se pueden eliminar.

NOTA: Estos valores no influyen en el rango de los datos de salida de *rand*, porque siempre va a estar entre 0 y 1. Por lo que pueden ser cualquier número. Incluso esas dos “semillas” pueden ser el mismo valor.

Y para conseguir que el valor esté comprendido entre 0 y 100, es necesario multiplicar por 100.0.

NOTA: el valor siempre tiene que ser decimal, si no lo es dará error.

Ahora al simular podemos ver que se están generando de forma correcta los valores aleatorios y además, el *integer* que tiene la señal “a” sirve como redondeo del valor de salida.



A partir de aquí se pueden generar tantos números aleatorios como simulaciones con estos se desee.

Código total

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use ieee.math_real.all;
entity test_tb is
end test_tb;
architecture Behavioral of test_tb is
signal a : integer;
signal b : real;
begin
   process
      variable seed1: integer:=0 ;
      variable seed2: integer:=99;
      variable rand : real;
   begin
      uniform(seed1, seed2, rand);
      b <= rand;
      a <= integer(rand*100.0);
      wait for 10ns;
   end process;
end Behavioral;
```


Nota final

Aunque este procedure pueda parecer muy aleatorio, es aleatorio hasta cierto punto. Porque si yo simulo con los mismos parámetros, siempre obtendré los mismos valores en el mismo instante de tiempo. Por ejemplo, en el ejemplo anterior, da igual si lo simulo 1 o 1000 veces siempre va a dar los mismos valores, y eso se puede ver que al comienzo de la simulación siempre aparecerán los mismos valores, esto es importante tenerlo en cuenta, ya que cambiando los valores de seed se pueden cambiar los valores aleatorios de salida del *procedure*.