Uso Correcto de los Paquetes IEEE para Matemática Simple

Nota Técnica 6

Cristian Sisterna

A fin de aclarar algunos conceptos del uso de paquetes con operaciones matemáticas simples, se debe saber que los siguientes paquetes nunca deberían usarse juntos en un mismo proyecto:

`ieee.numeric_std.all / ieee.std_logic_arith.all / ieee.unsigned.all/ ieee.signed.all;`

`ieee.numeric_std.all` se refiere a un paquete de la librería estandard de la IEEE; la que debería ser usada para todos los diseños nuevos.

`ieee.std_logic_arith.all`,`ieee.std_logic_unsigned.all`, `ieee.std_logic_signed.all` se refieren a un paquete definido por Synposys algunos años atrás, y que por ser una de las herramientas de síntesis más usadas, estos paquetes eran usados casi por defecto.

Hasta el día de hoy Xilinx ISE usa los siguientes paquetes, por defecto, cuando se usa la opción de crear un nuevo módulo VHDL:

```
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

A fin de que el código VHDL a escribir sea 100% IEEE estandard, estos paquetes deben ser reemplazados por:

```
use ieee.numeric_std.all;
```

Sin embargo, vale la pena aclarar lo siguiente: el paquete `std_logic_arith` es normalmente usado porque tiene definido operaciones matemáticas, como suma y resta, para el tipo `std_logic`, el `std_logic_vector` y el tipo `integer`. El paquete `numeric_std` NO tiene definidas operaciones matemáticas para `std_logic`, `std_logic_vector`, pero sí las tiene definidas para los tipos `signed`, `unsigned` e `integer`. Por ello para los casos en que sean necesarias operaciones matemáticas entre señales o variables, estas se deberán declarar como tipo `signed`, `unsigned` o `integer`, y usar 'casting' o funciones de conversión para pasar de `unsigned/signed/integer` a `std_logic_vector`.

El ejemplo más típico es el de un contador descrito en VHDL. La forma correcta de describirlo sería la siguiente:

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.numeric_std.all;
 4
 5 -- entity declaration
 6 entity counter_load is
 7    port (
 8         -- clock, reset ports
 9         clock           : in std_logic;
10         reset           : in std_logic;
11         -- control ports
12         load            : in std_logic;
13         -- input ports
14         initial_value   : in std_logic_vector(3 downto 0);
15         -- output ports
16         count           : out std_logic_vector(3 downto 0)
17  ) ;
18 end counter_load;
19
20 -- architecture declaration
21 architecture counter_load_beh of counter_load is
22  signal count_i: unsigned(3 downto 0);
23
24 -- architecture body
25 begin
26    count_proc: process(clock, reset)
27   begin
28     if(reset='0') then
29       count_i <= (others => '0');
30     elsif(rising_edge(clock)) then
31       if (load = '1') then
32         count_i <= unsigned(initial_value);
33       else
34         count_i <= count_i + 1;
35       end if;
36     end if;
37   end process count_proc;
38
39   count <= std_logic_vector(count_i);
40
41 end architecture counter_load_beh;
```

En este ejemplo se puede ver el use de 'cast'. En línea 32 se usa el cast `unsigned` para asignar el `standard_logic_vector` initial_value al unsigned count_i. Mientras que en línea 39 se usa el cast `std_logic_vector` para asignar el `unsigned` count_i al standard_logic_vector count.

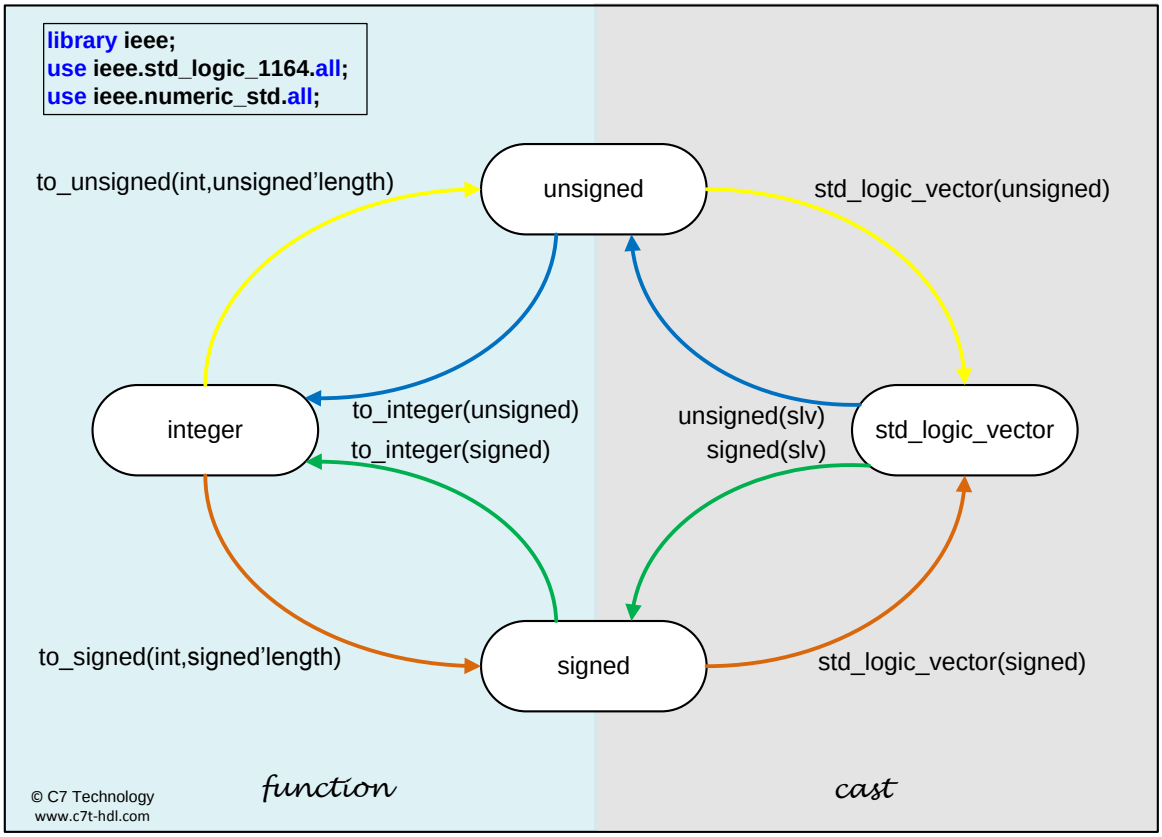
Otro ejemplo similar es el siguiente:

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.numeric_std.all;
 4
 5 -- entity declaration
 6 entity counter_ud is
 7    port (
 8         --  clocks, reset ports
 9         clock     : in std_logic;
10         reset     : in std_logic;
11         --  control ports
12         load    : in std_logic;
13         --  inputs ports
14         initial_value: std_logic_vector(3 downto 0);
15         --  output ports
16         count    : out std_logic_vector(3 downto 0)
17         );
18 end counter_ud;
19
20 -- architecture declaration
21 architecture counter_ud_beh of counter_ud is
22   signal count_i: integer range 0 to 15;
23
24 -- architecture body
25 begin
26   count_proc: process(clock, reset)
27   begin
28     if(reset='0') then
29       count_i <= 0;
30     elsif(rising_edge(clock)) then
31       if(load = '1') then
32         count_i <= to_integer(unsigned(initial_value));
33       else
34         count_i <= count_i + 1;
35       end if;
36     end if;
37   count <= std_logic_vector(to_unsigned(count_i,4));
38   end process count_proc;
39
40 end architecture counter_ud_beh;
```

En este caso en línea 32 ocurre algo interesante: primero se usa el cast `unsigned` para convertir a `unsigned` el `std_logic_vector` initial_value. Luego se usa la función `to_integer` para convetir el `unsigned` a `integer` y asignárselo a count_i, que lógicamente es de tipo integer. En línea 37 ocurre algo similar, pero no igual :) : se usa la función `to_unsigned` para convertir el entero count_i que se puede representar con 4 bits (count_i es declarado de 0 a 15), y luego se usa el cast `std_logic_vector` para pasar de `unsigned a std_logic_vector` y así poder asignar el vector a count que es un `std_logic_vector`.

Diagram de Funciones y Cast para conversión de tipos en VHDL

este blog describo en un simple diagrama cuales son las diferentes funciones de conversión de tipo disponibles en el paquete `numeric_std` y su modo de uso para las conversiones respectivas. También se detalla el uso de cast para conversiones entre tipos relacionados como `std_logic_vector` y `unsigned`.



El uso de este diagrama para la conversión de tipos es bastante sencillo.

Por ejemplo, para convertir un tipo `integer` a un tipo `std_logic_vector`, hay que seguir la línea amarilla junto con las funciones respectivas: se comienza con la función `to_unsigned(int, unsigned'length)` y luego se usa el cast `std_logic_vector` sobre el unsigned obtenido previamente. `unsigned'length` significa el número de bits necesarios para representar el respectivo integer. Por ejemplo 4 bits para el entero mayor que 7 y menor o igual que 15; 3 bits para un entero entre 0 y 7; etc.

Para ir del tipo `std_logic_vector` al tipo `signed`, solo hace falta aplicar el cast `signed` sobre el slv (std_logic_vector), es decir la línea verde entre ambos.