Conceptos Avanzados de VHDL

Hipólito Guzmán Miranda
Profesor Contratado Doctor
Universidad de Sevilla
hguzman@us.es

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

El VHDL que conocéis (síntesis)

```
comb: process
    if … elsif ... else … end if;
    case … when => ... end case;

sinc: Process
    if (rst = '1') then  ...
    elsif (rising_edge(clk) then ...
    end if;
```

Instancias de componentes

El VHDL que conocéis (simulación)

clk_process: **process**

- invertir clk, **wait for** clk_period/2

stim_process: **process**

- Secuencia de estímulos generada manualmente (muy tedioso en tests complejos)

¿Por qué?

‘Cargo cult programming’

En C, código descuidado normalmente produce malos resultados, y es más difícil de depurar y modificar

En VHDL, código descuidado puede producir hardware que funcione, pero también será difícil de depurar y modificar -> **código que nadie quiere tocar**

¿Por qué?

VHDL es un lenguaje de ALTO nivel

- Describid a un mayor nivel de abstracción
- Dejad que el sintetizador infiera el circuito
- El hardware sintetizado funciona igual de bien (o mejor), pero el código es más sencillo de leer y mantener

Pero vayamos poco a poco...

¿Por qué?

Unas palabras de advertencia

Todo lo que se explica aquí cuesta recursos hardware (en implementación)

Las operaciones no se realizan secuencialmente sino concurrentemente

Paradigma de diseño es el mismo: operaciones se convierten en lógica -> pero tendréis más recursos para estructurar vuestro código

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

El tipo de dato record

"Bad programmers worry about the code. Good programmers worry about data structures and their relationships." - Linus Torvalds

- Esto también aplica cuando describimos hardware
- Los records son un tipo de dato que está compuesto de otros datos
- Son el equivalente VHDL a los 'structs' de C
- Agrupad señales o puertos del mismo contexto en records

Ejemplo: signals

```
type transceiver_data is
    record
      data : std_logic_vector (15 downto 0);
      valid : std_logic;
    end record;

signal datain, dataout : transceiver_data;
```

Ejemplo: puertos

```
entity transceiver is
    Port (
      clk          : in std_logic;
      rst          : in std_logic;
      data_in      : in transceiver_data;
      data_out_I   : out transceiver_data;
      data_out_Q   : out transceiver_data
    );
end transceiver;
```

Ejemplo: puertos

```
entity transceiver is
```

- El record entero tiene que tener una única dirección (IN o OUT)
- Añadir una nueva señal al puerto sólo implica cambiar la definición del record!

```
        data_out_Q : out transceiver_data
    );
end transceiver;
```

Asignación y uso

Acceder a **record.dato** :

```
if (data_in.valid = '1') then
    data_out.data <= data_in.data;
    data_out.valid <= '1';
end if;
```

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

Leer señales, devolver valor

```
function invert (data: std_logic) return std_logic is
  begin
    return not data;
  end function invert;
```

Cada llamada a la función generará un inversor al sintetizar!

Ejemplos

```
function sum (a: integer; b: integer)
return integer is
  begin
    return a+b;
  end function sum;
```

Cada llamada a la función generará un sumador al sintetizar

Ejemplos

```
function sel (cond: boolean; if_true,
if_false: integer) return integer is
  begin
    if cond = true then
      return (if_true);
    else
      return (if_false);
    end if;
  end function sel;
```

¿Por qué usarlas?

Ya que producen el mismo hardware, merece la pena usarlas para:

- Encapsular operaciones que reutilizas
- Multiplexar o invertir generics, constants o señales en un **`generic map`** o **`port map`**

Insisto: no son subrutinas, son **hardware**!

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

Múltiples entradas, múltiples salidas

Aparentemente similares a los `functions` pero:

- Tienen parámetros IN y OUT
- Pueden leer de los IN y modificar los OUT

Ejemplo:

```
procedure vect_write
(constant data: in std_logic_vector(31 downto 0);
signal vector_ctrl : out fifo_ctrl) is
  begin
    vector_ctrl.datai <= data;
    vector_ctrl.wr_en <= '1';
    wait for 10 ns;
    vector_ctrl.wr_en <= '0'; --after 10 ns;
  end procedure;
```

(No es sintetizable ya que contiene un wait)

Diferencias:

- Las funciones no modifican nada, simplemente devuelven un valor

```
data <= a_function (other_data);
```

- Los procedimientos cambian el valor de señales

```
my_procedure (signals_in, signals_out);
```

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

Bucle for

```
for i in 0 to 7 loop
```

- El sintetizador **expande el bucle durante la síntesis**
- El rango del bucle debe ser estático (para que pueda sintetizarse)
- Cada paso por el bucle no es una ‘iteración’, sino una repetición del hardware

For loop

Ejemplo

```
reorder_data: process (data_in)
begin
  for i in 0 to 7 loop
    data_out(i) <= data_in(7-i);
  end loop;
end process;
```

For loop

Es equivalente a:

```
reorder_data: process (data_in)
begin
  data_out(0) <= data_in(7);
  data_out(1) <= data_in(6);
  data_out(2) <= data_in(5);
  data_out(3) <= data_in(4);
  data_out(4) <= data_in(3);
  data_out(5) <= data_in(2);
  data_out(6) <= data_in(1);
  data_out(7) <= data_in(0);
  end loop;
end process;
```

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

Instanciación condicional de componentes

- Instancia un componente o no, según se cumpla o no una condición
- Esta condición debe ser estática de forma que se sepa si se cumple **durante la síntesis**
- El sintetizador es el que instancia o no el componente

if condition generate

```
second_instance: if GENERATE_TWO=true
generate
  inst2: cont port map (
    clk => clk,
    rst => rst,
    count => count2 );
end generate second_instance;
```

Instanciación múltiple de componentes

- Usando **`for parameter in range`**
- Al igual que antes, el sintetizador **expande el bucle durante la síntesis**
- El rango del bucle debe ser estático (para que pueda sintetizarse)
- Cada paso por el bucle no es una 'iteración', sino una **instancia** del componente

Sentencia generate

for parameter in range generate

```
regdesp:
  for i in 0 to 3 generate
    myreg : reg port map (
      clk => clk,
      rst => rst,
      din => data(i),
      dout => data(i+1));
   end generate regdesp;
```

```
channel_filter: for i in 0 to N-1 generate
  taps: tap generic map(
      INPUT_WIDTH    => 9,
      OUTPUT_WIDTH   => 10,
      TRUNC_BITS     => 8,
      COEF           => coefs(sel(i<12, i, 23-i)),
      SAT_MULT_BITS => 2)
    port map(
      clk => clk,
      rst => rst,
      valid => cfilterin_valid,
      input => cfilterin,
      prev => d_aux(i),
      output => d_aux(i+1)
    );
  end generate;
```

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

Encapsular todo lo anterior

En un **package** VHDL podemos definir:

- Tipos de datos
- Constantes
- Funciones
- Procedimientos
- Componentes

Encapsular todo lo anterior

En lugar de redeclarar todo lo que necesitemos en cada vhd de cada entidad, simplemente añadimos a la sección `library`:

```
use work.mypackage.all;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package mypackage is
    -- declaración de tipos de datos
    -- declaración de constantes
    -- declaración de componentes
    -- declaración de funciones y procedimientos
end mypackage;

package body mypackage is
    -- definición de funciones y procedimientos




end mypackage;
```

VHDL avanzado

- ¿Por qué VHDL avanzado?
- Records
- Functions
- Procedures
- For loop
- Sentencia Generate
- Packages
- Libraries

Conjuntos de packages

Se incluyen por completitud, pero no os harán falta crearlas para la asignatura

Por ejemplo, std_logic_1164 es un package del library IEEE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
```

Vuestros packages custom pertenecen al library **`work`** por defecto

Conclusiones y recomendaciones

- VHDL da opciones para estructurar el código de manera que sea más mantenible
- No es obligatorio usarlo todo
- Aunque esté encapsulado, sigue siendo HW!
- Se recuerda que Xilinx ISE genera plantillas para todo lo mencionado anteriormente (en Project -> New source y en Edit -> Language Templates)