Lenguajes de Descripción de Hardware

VHDL

Fernando Olivera Domingo

Contenidos

- VHDL
- Herramientas de Xilinx

VHDL

- **HDL (Lenguajes de Descripción de Hardware)**
  - Lenguajes creados para documentar las interconexiones y funcionamiento de un circuito electrónico.
  - Utilizados para el diseño de ASIC y la descripción del *hardware* de las FPGA.
  - Lenguajes:
    - ►VHDL
    - ►Verilog
    - ►ABEL
  - Lenguajes parecidos a C o ADA, pero no son lenguajes de programación, sino de descripción de *hardware*.
    - ►Pensar en señales (cables) y no siempre en variables.
  - Very High Speed Integrated Circuits (VHSIC) + Hardware Description Language (HDL) ---> VHDL

VHDL

VHDL

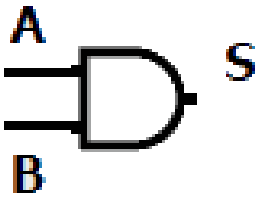
- Lenguaje formal, no tiene ambigüedades y esta normalizado. Es textual.
- Creado en 1987, convirtiéndose en la norma IEEE-1076.
- Las herramientas de síntesis lógica traducen la descripción textual en circuitos lógicos.
- En cierta medida la descripción en VHDL es independiente de la tecnología.
- Tres niveles de abstracción que se pueden combinar:
  - ►Flujo de datos (Dataflow)
  - ►Comportamiento (behaviour)
  - ►Arquitectural (structural)

VHDL

- **Dos aspectos caracterizan un circuito o sistema en VHDL:**
  - ENTITY
    - ►Es el interfaz externo. Define el nombre del circuito o sistema y el número y tipo de entradas y salidas del mismo
  - ARCHITECTURE
    - ►Es el algoritmo interno. La forma en que se procesan las señales de entrada y se generan las señales de salida.

```
entity and2 is
      port (
            A,B: in STD_LOGIC;
            S: out STD_LOGIC
      );
end and2;
```

```
architecture and2arch of and2 is
begin
      S <= A and B;
end and2arch;
```



VHDL

- **Sintaxis de Entity**

```
entity {nombre del circuito o sistema} is
        generic (
                {constantes}: {tipo} := {valor};
        );
        port (
                {señales de entrada}: in {tipo de dato};
                {señales de salida}: out {tipo de dato};
                {señales bidireccionales para triestado}: inout {tipo de dato};
                {señales de salida para realimentación}: buffer {tipo de dato}
        );
end  {nombre del circuito o sistema};
```

VHDL

Sintaxis de Architerture

```
architecture {nombre de la arquitectura} of {nombre de la entidad} is
    constant {nombre constante} : {tipo de dato} := {valor};
    variable {nombre variable} : {tipo de dato} := {valor opcional};
    signal {nombre señal interna} : {tipo de dato};

    begin

        {zona de descripción}; -- Todo se ejecuta en paralelo

    end {nombre de la arquitectura}
```

VHDL

- **Librerias**
  - Definidos en librerias.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Para utilizar el tipo std_logic
use  IEEE.NUMERIC_STD.ALL;
-- Para utilizar los tipos signed y unsigned que admiten operaciones
-- aritmético-lógicas
use  IEEE.STD_LOGIC_ARITH.ALL;
-- Para utilizar las operaciones aritmético-lógicas
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- Si los vectores std_logic están en representación binaria pura
use IEEE.STD_LOGIC_SIGNED.ALL;
-- Si los vectores std_logic están en representación complemento a 2
```

VHDL

- **Tipos de datos**
  - Bit: '0' o '1'.
  - Bit_vector: vector de bits. Hay que definir rango.
  - Boolean: true o false.
  - Character: valores ASCII.
  - String: cadenas ASCII. Hay que definir rango.
  - Integer: Enteros. Hay que definir rango.
  - Natural: Naturales. Hay que definir rango.
  - Positive: Positivos. Hay que definir rango.
  - Real: Reales. Hay que definir rango.
  - Std_logic: '0', '1', 'Z', 'U', 'X', 'W', 'L', 'H', '-'.
  - Std_logic_vector: vector de std_logic. Hay que definir rango.

VHDL

Tipos de datos

- Todos los puertos de entity tienen que ser de tipo std_logic o std_logic_vector para que el diseño sea sintetizable. Esto es porque estos tipos representan una señal más real: tienen estados adicionales de alta impedancia, desconocido...

- No se puede olvidar que estamos describiendo hardware y no programando software, por lo que vamos a trabajar con señales que corresponderían con cables de un circuito. Señales como natural o integer también van a ser sintetizadas como señales con el número necesario de bits para una representación completa.

VHDL

- **Tipos de datos**
  - Tipo enumerado

    type {nombre} is ({valor1},{valor2},...);
  - Array

    type {nombre} is array {rango} of {tipo};
  - Record

    type {nombre} is record

    {elemento1} : {tipo de dato 1};

    {elemento2} : {tipo de dato 2};

    end record;
- **Asignaciones**
  - <= Asignación de señales
  - := Asignación de variables

VHDL

Tipo	Datos de Entrada	Devuelve	Operadores
Lógicos	std_logic, std_ulogic, std_logic_vector, std_ulogic_vector	std_logic, std_ulogic, std_logic_vector, std_ulogic_vector	Not, and, or, nand, nor, xor
Relación	Cualquiera	Boolean	=, /=, <, <=, >, >=
Aritméticos	integer, positive, natural, std_logic_vector, std_ulogic_vector	integer, positive, natural, std_logic_vector, std_ulogic_vector	+, -
Concatenación	Cualquiera (homogéneos)	Array de elementos	&

VHDL

- **Combinacional y secuencial**
  - Combinacional

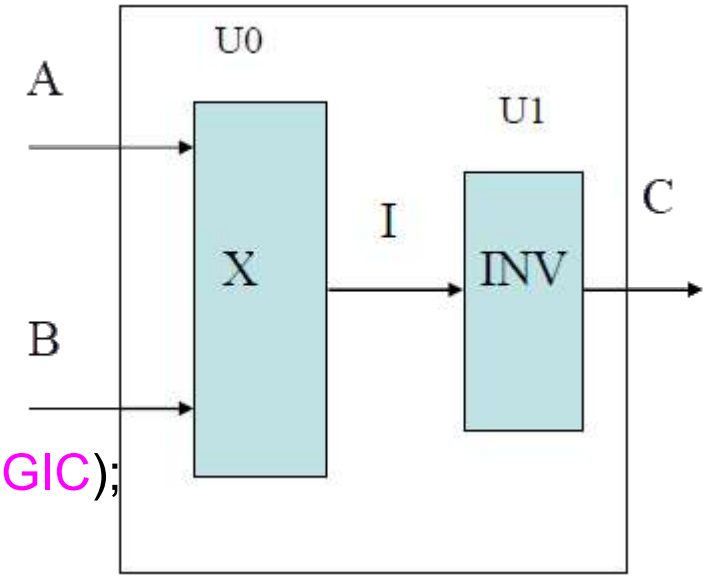
```
C <= not(A xor B);
D <= '1' when c='0' else
     '0' when c='1';
```

  - Secuencial

```
process (A,B)
begin
     if (A = B) then
          C <= '1';
     else
          C <= '0';
     endif;
end process;
```

VHDL

- **Componentes**




```
architecture estructural of ejemplo is begin
      signal I: STD_LOGIC;
      component X
            port (x,y: in STD_LOGIC; z : out STD_LOGIC);
      end component;
      component INV
            port (e: in STD_LOGIC; s : out STD_LOGIC);
      end component;
begin
            U0: X port map (A,B,I);
            U1: INV port map (I,C);
end estructural;
```

VHDL

When – else

{señal a modificar} <= {valor 1} when {condición 1} else
{valor 2} when {condición 2} else
...
{valor n} when {condición n} else
{valor por defecto};

- Ejemplo

C <= “00” when A=B else
“01” when A <B else
“10”;

VHDL

With – select – when

```
with {señal condición} select
{señal a modificar} <= {valor 1} when {valor 1 condición},
                       {valor 2} when {valor 2 condición},
                       ...
                       {valor n} when {valor n condición},
                         {valor por defecto} when others;
```

- Ejemplo

```
with entrada select
salida <= "00" when "0001",
          "01" when "0010",
          "10" when "0100",
          "11" when others;
```

VHDL

- **If – then - else**

```
process ({lista de sensibilidad})
begin
if {condición} then
      -- Asignaciones;
elsif {otra condición} then
      -- Asignaciones;
else
      -- Asignaciones;
end if;
end process;
```

- Ejemplo

```
process (control, A, B)
begin
if control = “00” then
      resultado <= A + B;
elsif control = “11”  then
      resultado <= A – B;
else
      resultado <= A;
end if;
end process;
```

VHDL

Case - when

```
process ({lista de sensibilidad})
begin
case {señal condición} is
        when {valor 1 condición} =>
        -- Asignaciones;
        when {valor 2 condición} =>
        -- Asignaciones;
        when others =>
        -- Asignaciones;
end case;
end process;
```

- Ejemplo

```
process (control, A, B)
begin
case control is
        when “00” =>
            resultado <= A + B;
        when “11” =>
            resultado <= A - B;
        when others =>
            resultado <= A;
end case;
end process;
```

VHDL

- **For – loop**

```
process ({lista de sensibilidad})
begin
for {variable del bucle} in {rango} loop
      -- Asignaciones
end loop;
end process;
```

  - Ejemplo

```
process (A)
begin
for i in 0 to 7 loop    -- Podría ser 7 downto 0
      B (i+1) <= A(i);
end loop;
end process;
```

VHDL

- **While – loop**

```
process ({lista de sensibilidad})
begin
while {condición} loop
      -- Asignaciones
end loop;
end process;
```

- Ejemplo

```
process (A)
variable i: natural :=0;
begin
while i <7 loop
      B (i+1) <= A(i);
      i := i+1;
end loop;
end process;
```

Herramientas de Xilinx

- **Son necesarias algunas herramientas para programar en la Spartan 3 nuestros diseños en esquemáticos o VHDL.**
  - Xilinx ISE o Xilinx Webpack
    - ►Herramienta para realizar esquemáticos y código VHDL, que nos permite sintetizar, traducir, implementar y simular nuestros diseños además de programarlos con un cable JTAG.
    - ►Xilinx Webpack es la herramienta gratuita, mientras que Xilinx ISE es la de pago. Son muy similares salvo porque con la primera están limitados los modelos de FPGA que pueden ser utilizados.

Herramientas de Xilinx

- **Xilinx ISE**
  - Instrucciones básicas para desarrollar en las FPGA de Xilinx.
  - Xilinx ISE es una de las herramientas de Xilinx para FPGAS, se utiliza para diseño lógico.
  - Xilinx tiene otras aplicaciones para el procesamiento embebido o el procesamiento de señal digital.

Herramientas de Xilinx

- Xilinx ISE o Webpack