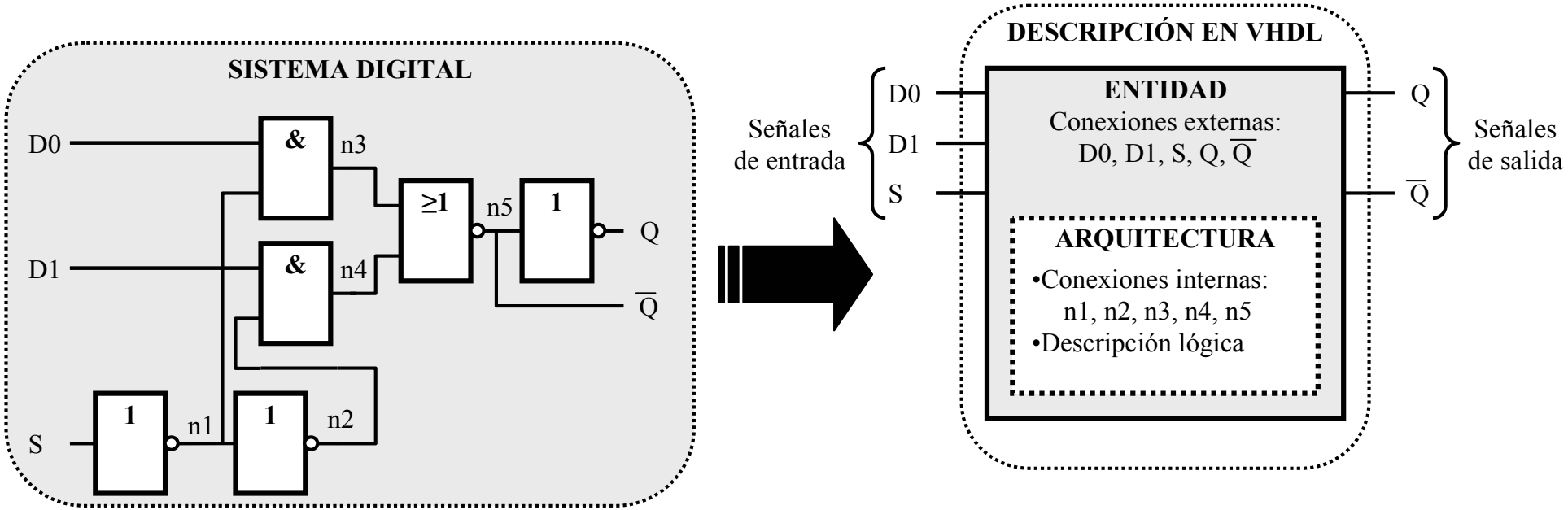
ELECTRÓNICA DIGITAL

Tema 17

LENGUAJES DE DESCRIPCIÓN DE LOS SISTEMAS DIGITALES (PARTE 2)

LENGUAJE VHDL

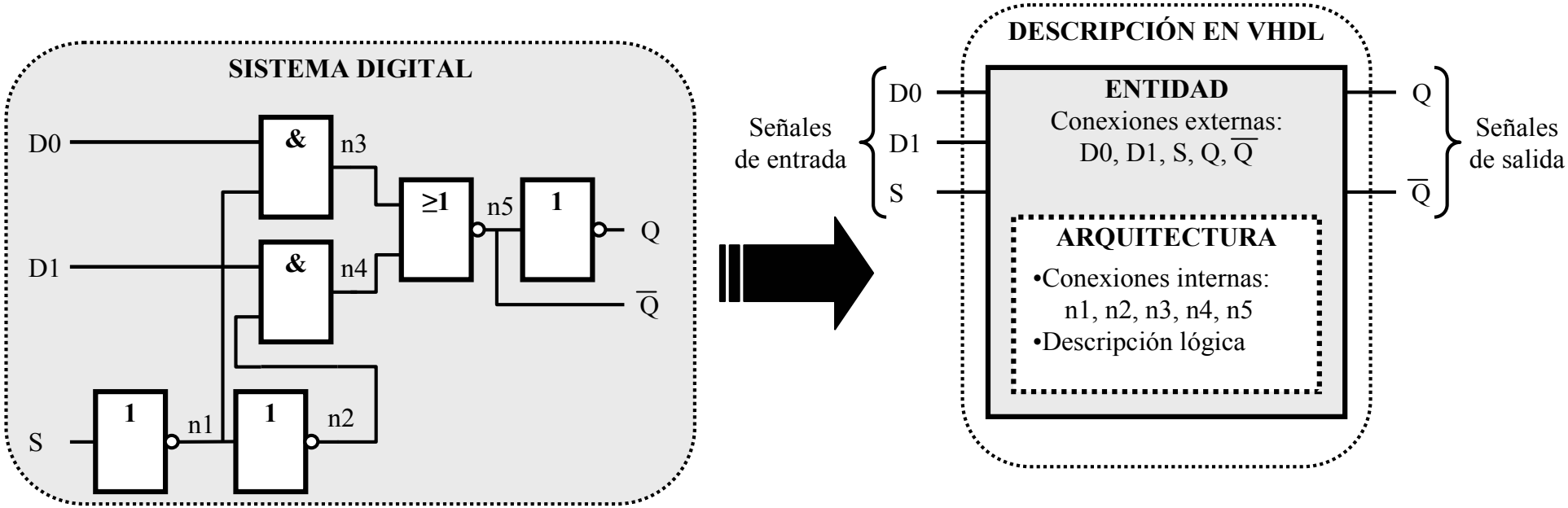
Describe cualquier sistema digital mediante la definición de un modelo que proporciona una descripción externa y una o más descripciones internas del mismo. Como ejemplo, en la figura se muestra la descripción en VHDL de un multiplexor de dos canales. La descripción externa se realiza mediante el concepto de entidad (*ENTITY*) y la interna mediante el de arquitectura (*ARCHITECTURE*).



LENGUAJE VHDL

ENTIDAD (*ENTITY*)

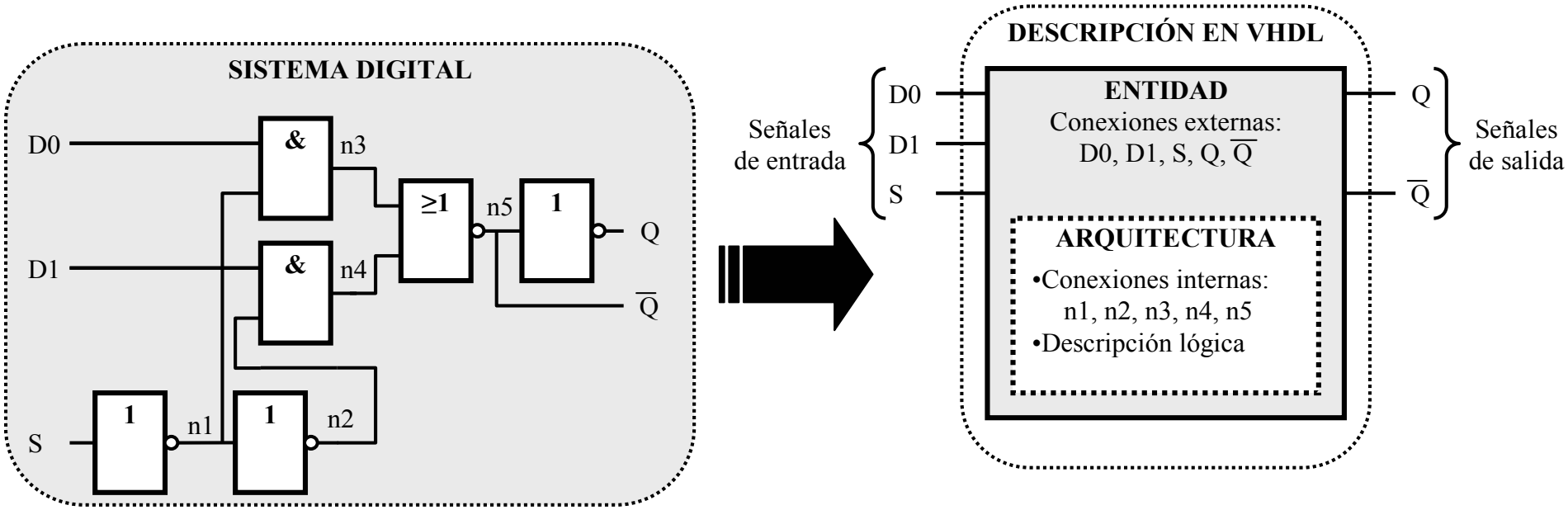
La entidad corresponde a la descripción externa y está relacionada con los terminales de entrada y salida que permiten al sistema digital interactuar con el exterior. Cada sistema digital que se describe en VHDL constituye una entidad. El concepto de entidad facilita el diseño jerárquico de los sistemas digitales complejos, que se caracteriza porque una entidad de nivel superior se describe mediante la interconexión de entidades de nivel jerárquico inferior.



LENGUAJE VHDL

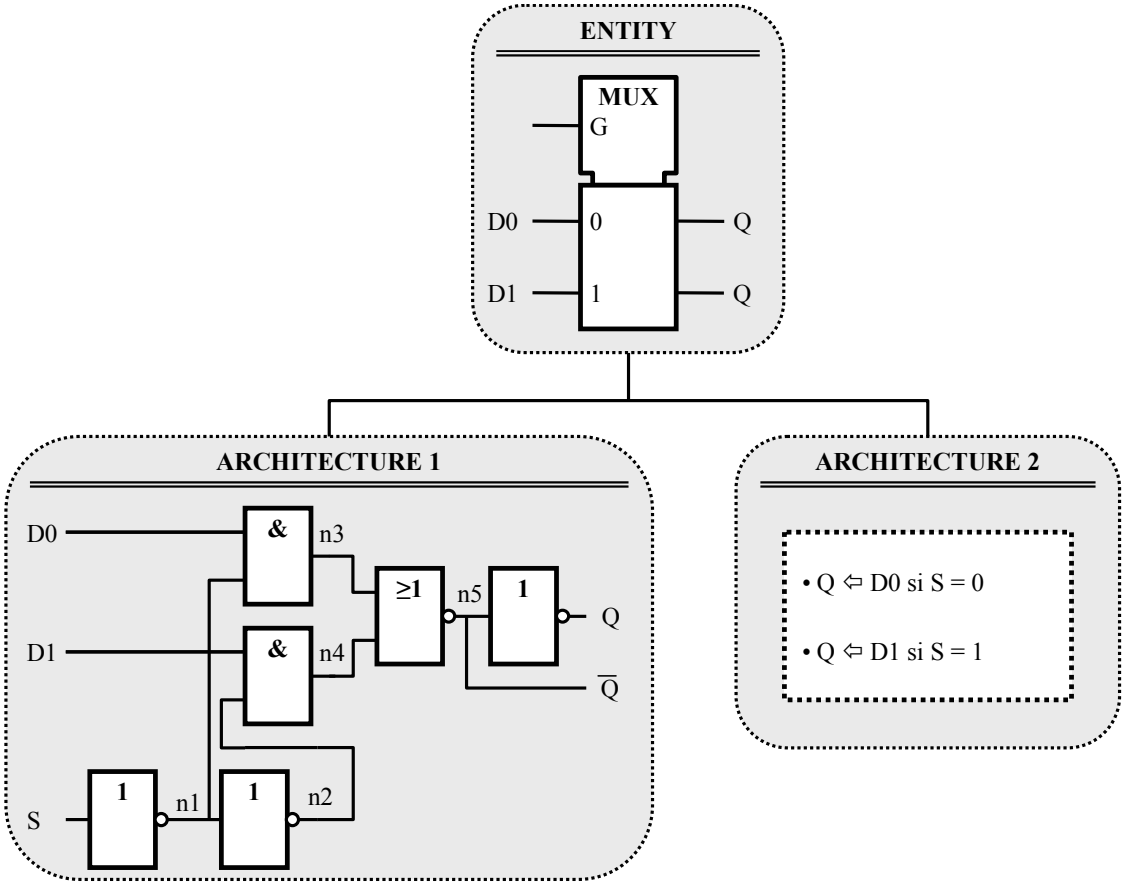
ARQUITECTURA (*ARCHITECTURE*)

La arquitectura, por su parte, especifica el funcionamiento interno del sistema digital y se puede referir tanto a su estructura como a su comportamiento. Puesto que en la práctica totalidad de los casos existen diferentes formas de especificar el comportamiento o la estructura de un sistema digital, una misma entidad puede tener asociada una o más arquitecturas.



LENGUAJE VHDL
EJEMPLOS DE ARQUITECTURAS

La arquitectura 1 es una descripción estructural de la implementación con puertas lógicas *Y* y *NOR* e inversores, representada mediante su esquema. La arquitectura 2 es una descripción algorítmica de su comportamiento



LENGUAJE VHDL

ESTRUCTURA GENÉRICA DE UN FICHERO .vhdl

•La definición de las bibliotecas (*Libraries*) que se utilizan en la descripción.

•La definición de la entidad (*Entity*), que define el componente como una caja negra con conexiones externas.

•La arquitectura (*Architecture*) que está asociada a dicha entidad y constituye la descripción lógica del componente.

```
library nombre_de_biblioteca;                                   } Definición de
use nombre_de_biblioteca. nombre_de_paquete.componentes;        } las bibliotecas

entity nombre_de_entidad is                                     } Definición de
        ...( Zona de declaraciones )                            } la entidad
end nombre_de_entidad ;                                         }

architecture nombre_de_arquitectura of nombre_de_entidad is     }
        ...( Zona de declaraciones )                            } Definición de
begin                                                           } la arquitectura
        ...( Cuerpo de la arquitectura )                        }
end nombre_de_arquitectura;                                     }
```

LENGUAJE VHDL

Ejemplo de descripción en VHDL

```
-- Descripción mediante el flujo de datos en VHDL
-- de la puerta NAND de dos entradas
-- por Yago Mandado Mayo 2007
--------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
--------------------------------------------------------------------------------
-- Definición de la entidad de la puerta NAND de dos entradas
ENTITY puerta_NAND2entradas IS
 PORT (a,b: IN std_logic;
              f : OUT std_logic);
 END puerta_NAND2entradas;
--------------------------------------------------------------------------------
-
-- Definición de la arquitectura de flujo de datos
-- de la puerta NAND de dos entradas
ARCHITECTURE flujo_de_datos OF puerta_NAND2entradas IS
BEGIN
     f <= a NAND b;
 END flujo_de_datos;
```

CARACTERÍSTICAS DEL LENGUAJE VHDL

- **Es un lenguaje estructurado que facilita la realización de descripciones jerárquicas (multinivel) en las que un sistema digital se modela como un conjunto de componentes interconectados y cada componente, a su vez, como un conjunto de componentes de menor jerarquía interconectados (subcomponentes) y así sucesivamente.**

- **Permite la realización de diseños parametrizables mediante la utilización de parámetros genéricos.**

CARACTERÍSTICAS DEL VHDL

- **Soporta tres tipos diferentes de descripción o modelado de los sistemas digitales:**
  - **La descripción estructural (*`Structural modeling`*) en la que se especifican los componentes que forman el sistema digital y sus interconexiones.**
  - **La descripción algorítmica o de comportamiento (*`Behavioural modeling`*) en la que se especifica el algoritmo que define el funcionamiento del sistema digital.**
  - **La descripción de flujo de datos (*`Dataflow modeling`*), conocida también como de transferencia entre registros [(*`Register Transfer Level`* (RTL))], en la que se especifica el comportamiento de las señales de salida del sistema digital a partir de las señales de entrada.**
- **Facilita la descripción de un sistema digital mediante la combinación de diferentes tipos de descripciones lo que constituye una descripción mixta.**

CARACTERÍSTICAS DEL VHDL

CONCEPTOS GENERALES

- **Identificadores y palabras reservadas**
- **Comentarios**
- **Tipos de datos**
- **Objetos de datos**
- **Atributos**
- **Operadores predefinidos**
- **Componentes**

CONCEPTOS GENERALES DEL VHDL

IDENTIFICADORES

Nombre asignado a cada uno de los elementos (señales, variables, entidades, arquitecturas, etc.) que forman parte del VHDL. Está constituido por un conjunto de letras, números, y el carácter "_". El primer carácter de un identificador tiene que ser una letra y se pueden escribir indistintamente con letras mayúsculas o minúsculas. En la Tabla 9.7 se indican diversos ejemplos de identificadores.

flujo_de_datos
Multiplexor2canales
Entrada_a

CONCEPTOS GENERALES DEL VHDL

PALABRAS RESERVADAS

A	abs access after alias all and architecture array assert attribute
B	begin block body buffer bus
C	case component configuration constant
D	disconnect downto
E	else elsif end entity exit
F	file for function
G	generate generic group guarded
I	if impure inertial inout is
L	label library linkage literal loop
M	map mod
N	nand new next nor not null
O	of on open or others out
P	package port postponed procedure process pure
R	range record register reject rem report return rol ror
S	select severity shared signal sla sll sra srl subtype
T	then to transport type
U	unaffected units until use
V	variable
W	wait when while with
X	xnor xor

CONCEPTOS GENERALES DEL VHDL

COMENTARIOS (*COMMENTS*)

Tienen como objetivo facilitar la comprensión de una descripción

Deben estar precedidos de dos guiones, tal como se indica en la tabla. Si un comentario no cabe en una sola línea se pueden utilizar varias líneas, precedidas todas ellas de dos guiones.

```
-- Definición de la entidad multiplexor
-- Arquitectura de flujo de datos
-- Inicio del proceso
```

CONCEPTOS GENERALES DEL VHDL

TIPOS DE DATOS

Los tipos de de datos predefinidos en el lenguaje se encuentran en el paquete normalizado (*Standard*) de la biblioteca STD.
En la tabla se indican los principales tipos de datos predefinidos.
Para asignar valores a dichos datos se suele utilizar la base 10 y la notación de coma fija.

Tipo	Rango	Descripción
Integer	-MAXINT ... MAXINT	Números enteros
Natural	0 ...MAXINT	Números naturales
Positive	1 ...MAXINT	Números positivos
Real	-MAXREAL ... MAXREAL	Números reales (coma flotante)
Boolean	TRUE, FALSE	Valores lógicos
Bit	0, 1	Números binarios
bit_vector		Cadena de bit
Character	Véase paquete standard	Caracteres
String		Cadena de caracteres
Time	0 a MAXTIME fs	Unidades de tiempo (ps, ns, us, ms, sec, min, hr)

CONCEPTOS GENERALES DEL VHDL

OBJETOS DE DATOS (*DATA OBJECTS*)

Son elementos del lenguaje VHDL que almacenan información.

El hecho de ser un lenguaje de descripción de circuitos y sistemas digitales hace que el VHDL posea señales (*Signals*), además de las constantes (*Constants*) y las variables (*Variables*) que poseen los lenguajes de programación de alto nivel como por ejemplo el C [MAND 07].

A cada tipo de objeto de datos se le tiene que asignar uno de los tipos de datos indicados en el apartado anterior. A dicha asociación se la denomina declaración (*Declaration*).

CONCEPTOS GENERALES DEL VHDL

OBJETOS DE DATOS (*DATA OBJECTS*)

CONSTANTES

Son objetos de datos cuyo valor no cambia a lo largo de una descripción.

Se pueden declarar en una entidad o una arquitectura.

Debido a su carácter, las constantes se tienen que inicializar al declararlas.

Sintaxis de la declaración de las constantes:

CONSTANT lista_de_nombres: tipo [:= expresión];

```
CONSTANT  retardo: tiempo := 10 ns
CONSTANT A: std_logic_vector (3 DOWNTO 0) := "0110"
```

CONCEPTOS GENERALES DEL VHDL

OBJETOS DE DATOS (*DATA OBJECTS*)

SEÑALES (*SIGNALS*)

Son uno de los elementos más utilizados en VHDL porque sirven para aplicar valores a las entradas de un circuito, para obtenerlas a su salida y transmitirlas entre los elementos que lo componen. Las señales están intrínsecamente ligadas al carácter de lenguaje de descripción de circuitos digitales (*Hardware*) del VHDL y representan las conexiones o terminales físicos del mismo y por tanto poseen diversas propiedades que modelan el comportamiento de las conexiones reales.

Antes de utilizar una señal es necesario también asignarle un tipo de dato, es decir declararla. La sintaxis de la declaración de una señal es la siguiente:

SIGNAL lista_de_nombres: tipo [:= expresión];

en la que el carácter ';' indica el final de la declaración.

Por ejemplo la declaración de la señal de entrada *a* que solo puede estar en nivel uno o cero es:

SIGNAL *a*: **IN** bit;

CONCEPTOS GENERALES DEL VHDL

OBJETOS DE DATOS (*DATA OBJECTS*)

VARIABLES (*VARIABLES*)

Son datos cuyo valor puede cambiar a lo largo del funcionamiento. En la tabla se indican dos ejemplos.

```
VARIABLE registro: INTEGER;
VARIABLE amplificador: std_logic_vector (7 DOWNTO 0);
```

Sólo se pueden definir dentro de un proceso y quedan restringidas a él. Al definir una variable se le puede asignar un valor inicial y en caso contrario se le asigna un valor por defecto, como por ejemplo '0' para el caso de datos de tipo bit.

CONCEPTOS GENERALES DEL VHDL

ATRIBUTOS

Los atributos (*`Attributes`*) establecen características de algún elemento (entidad, arquitectura, señal, etc.) de una descripción en VHDL. Pueden ser de distinta índole como por ejemplo, valor, función, tipo, rango, etc. El VHDL posee un gran número de atributos predefinidos. Además permite la definición de atributos por parte del diseñador de sistemas digitales.

Un ejemplo de atributo aplicado a las señales es '*`event`*' que proporciona un valor verdadero (uno) si se produce un cambio de nivel de la señal, lo que constituye un suceso (*`Event`*) de la misma. Por ejemplo la expresión:

C' event AND C = '1'

toma el valor uno si se produce un flanco de subida de la variable C.

CONCEPTOS GENERALES DEL VHDL

OPERADORES PREDEFINIDOS

El lenguaje VHDL tiene también predefinidos un conjunto de operadores (*`Operators`*) divididos en tres grupos:

- **Aritméticos**
- **Relacionales**
- **Lógicos**

Especial atención merecen los operadores lógicos que corresponden a las funciones lógicas básicas. Dichos operadores se pueden combinar para describir cualquier expresión lógica.

CONCEPTOS GENERALES DEL VHDL

Al indicar una expresión lógica en VHDL se debe tener en cuenta que las reglas de prioridad de las operaciones del álgebra de Boole no se cumplen en VHDL. En VHDL la operación lógica *Y* y la operación *O*, tienen la misma prioridad y se ejecuta antes la que está más a la izquierda. Por ejemplo expresión VHDL "*a* OR *b* AND *c*" indica que se realiza la suma lógica de las variables *a* y *b*, y a continuación se realiza el producto lógico del resultado obtenido por la variable *c*.

•Para cambiar la prioridad de la ejecución de las operaciones lógicas en VHDL se utilizan los paréntesis. Por ejemplo la expresión en VHDL que realiza la función "*a* + *b c*" es:

•*a* OR (*b* AND *c*)

OPERADORES LÓGICOS PREDEFINIDOS

Operación	Operador
Y	AND
Y invertida	NAND
O	OR
O invertida	NOR
O-exclusiva	XOR
O-exclusiva invertida	XNOR
Inversión	NOT

CONCEPTOS GENERALES DEL VHDL

COMPONENTE

Cualquier descripción de un sistema digital constituida por una entidad y una arquitectura asociada con ella es un componente (COMPONENT) porque puede ser considerada como un elemento que forma parte de otro sistema digital más complejo.

Para que el conjunto formado por una entidad y una arquitectura pueda ser utilizado como componente es necesario declararlo como tal de forma explicita. Por ejemplo para utilizar una puerta NAND de dos entradas como componente de una arquitectura estructural hay que colocar en ella la declaración indicada en la tabla.

```
COMPONENT NAND2entradas IS
PORT (a,b: IN std_logic; f: OUT std_logic);
END COMPONENT;
```

ELEMENTOS BÁSICOS DE LA DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

- **Bibliotecas**

- **Entidades**

- **Arquitecturas**

ELEMENTOS BÁSICOS DE LA DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

BIBLIOTECA (*LIBRARY*)

Conjunto de elementos (componentes, tipos de datos, constantes, etc.) que se pueden utilizar al describir cualquier sistema digital.

El usuario puede establecer sus propias bibliotecas, pero hay bibliotecas normalizadas que contienen los elementos utilizados en la mayoría de las descripciones.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

ELEMENTOS BÁSICOS DE LA DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

ENTIDAD

Comienza con la palabra ENTITY (Entidad) seguida del nombre que se le asigna (que corresponde al sistema que se describe). A continuación, mediante la palabra PORT (puerto), se especifican las señales de entrada y salida del sistema así como su tipo y algún otro parámetro de interés, como por ejemplo, el tiempo de propagación de las señales a través del sistema. Finaliza con la palabra END seguida del nombre de la propia entidad.

```
-- Definición de la entidad puerta
-- NAND2entradas
ENTITY puerta_NAND2entradas IS
PORT (a, b              : IN std_logic;
      f                 : OUT
std_logic);
END puerta_NAND2entradas;
```

ELEMENTOS BÁSICOS DE UNA DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

ARQUITECTURA

Se inicia con el identificador ARCHITECTURE seguido del nombre que se le asigna y del de la entidad a la que se refiere. Al igual que en el caso de la entidad, para finalizar la definición de la arquitectura se utiliza la cláusula END seguida de su nombre.

Las arquitecturas tienen dos partes bien diferenciadas:

- **Una *zona de definiciones***

  En ella se especifican las distintas señales, constantes, componentes y otros elementos a utilizar.

- **El *cuerpo de la arquitectura*, en el que se describe el funcionamiento o la estructura de la entidad.**

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

FORMAS DE DESCRIBIR UNA ARQUITECTURA

- **Descripción de flujo de datos (*`Dataflow modeling`*)**

- **Descripción algorítmica o de comportamiento (*`Behavioural modeling`*)**

- **Descripción estructural (*`Structural modeling`*)**

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN DE FLUJO DE DATOS

Es idónea cuando se conocen las ecuaciones lógicas de la función que hay que implementar. Por ello es adecuada para describir los circuitos combinacionales. Facilita la descripción de los circuitos combinacionales a través de su expresión algebraica tal como se muestra en la página siguiente.

El flujo de datos también se realiza mediante instrucciones de asignación condicional de valores a las señales (*`Conditional signal assignment`*) de un circuito y permite realizar así la descripción de distintos bloques funcionales digitales tal como se realiza posteriormente.

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

EJEMPLO DE DESCRIPCIÓN DE FLUJO DE DATOS

```
-- Descripción de flujo de datos en VHDL
-- de la puerta NAND de tres entradas
-- por Yago Mandado Mayo 2007
----------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-----------------------------------------------------
-- Definición de la entidad de la
-- puerta NAND de tres entradas
ENTITY puerta_NAND3entradas IS
PORT (a, b, c: IN std_logic;
                    f : OUT std_logic);
END puerta_NAND3entradas;
-------------------------------------------------------
-- Definición de la arquitectura de flujo de datos
-- de la puerta NAND de tres entradas
ARCHITECTURE flujo_de_datos OF puerta_NAND3entradas IS
BEGIN
    f <= NOT (a AND b AND c);
END flujo_de_datos;
```

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN DE FLUJO DE DATOS

Asignación condicional de señales

Instrucción "WHEN- ELSE"

Esta instrucción indica que cuando (WHEN) se cumple una condición se realiza una asignación y en caso contrario (ELSE) se realiza otra o ninguna (NULL). Se utiliza para asignar valores a una señal.

```
SEÑAL<=     valor_1 WHEN condición_1 ELSE
            valor_2 WHEN condición_2 ELSE
            ...,
            ...,
            valor_n WHEN condición_n;
```

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN DE FLUJO DE DATOS

Ejemplo de utilización de la instrucción "WHEN- ELSE"

```
-- Descripción de flujo de datos en VHDL
-- de un multiplexor de dos canales de entrada
-- con la instrucción "WHEN-ELSE"
-- por Yago Mandado Mayo 2007
-----------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-----------------------------------------------------------------------
-- Definición de la entidad de un multiplexor de dos canales
ENTITY multiplexor_dos canales IS
 PORT (D0, D1, S          : IN std_logic;
          Q               : OUT std_logic);
 END multiplexor_dos canales;
 ------------------------------------------------------------------------
-- Definición de la arquitectura de flujo de datos
-- de un multiplexor de dos canales
-- con la instrucción "WHEN-ELSE"
ARCHITECTURE flujo_de_datos OF multiplexor_dos canales IS
BEGIN
     Q <= D0 WHEN (S = '1') ELSE
            D1 WHEN (S = '0');
 END flujo_de_datos;
```

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN DE FLUJO DE DATOS

Asignación condicional de señales

Instrucción "WITH- SELECT"

Esta instrucción asigna a una señal un valor en función del nivel lógico de una variable o de una expresión colocada entre las palabras reservadas WITH y SELECT.

```
WITH expresión SELECT
señal <=valor_1 WHEN resultado_1,
        valor_2 WHEN resultado_2,
        ...,
        valor_n WHEN resultado_n;
```

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN DE FLUJO DE DATOS

Ejemplo de utilización de la instrucción "WITH-SELECT"

```
-- Descripción de flujo de datos en VHDL
-- de un multiplexor de dos canales de entrada
-- con la instrucción "WITH-SELECT"
-- por Yago Mandado Mayo 2007
-------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-------------------------------------------------------------------
-- Definición de la entidad de un multiplexor de dos canales
ENTITY multiplexor_dos_canales IS
 PORT (D0,D1,S          : IN std_logic;
          Q             : OUT std_logic);
 END multiplexor_dos canales;
 ------------------------------------------------------------------------
-- Definición de la arquitectura de flujo de datos
-- de un multiplexor de dos canales
-- con la instrucción "WITH-SELECT"
ARCHITECTURE flujo_de_datos OF multiplexor_dos canales IS
BEGIN
        WITH (S) SELECT
                Q <= D0 WHEN '1';
                     D1 WHEN '0';
 END flujo_de_datos;
```

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN ALGORÍTMICA

El mecanismo más utilizado para describir el comportamiento de un sistema digital mediante un algoritmo es el proceso (*PROCESS*) que está compuesto por un conjunto de instrucciones que se ejecutan secuencialmente. Un proceso puede a su vez ser ejecutado concurrentemente con otros. La sintaxis de un proceso es la indicada en la tabla.

La lista de sensibilidad es una relación de señales cuyos cambios de estado activan la ejecución del proceso (en esta lista no se pueden incluir variables ni constantes). Es posible sustituir la lista de sensibilidad mediante la inclusión de la instrucción

WAIT dentro del proceso

[etiqueta :] **PROCESS** [(lista de sensibilidad)] [**IS**] -
- Zona de declaraciones
BEGIN
- - Cuerpo del proceso
- - (Sólo instrucciones secuenciales)
END PROCESS [etiqueta];

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN ALGORÍTMICA

Instrucciones secuenciales: Instrucción "WAIT"

Es una de las más utilizadas en VHDL para describir circuitos secuenciales síncronos. Se caracteriza por detener la ejecución de un proceso hasta que se cumple una determinada condición y su sintaxis se representa en la tabla.

```
[ etiqueta : ]
WAIT [ON señal_1, señal_2, ..]
        [UNTIL condición]
        [FOR expresión_temporal];
```

A continuación se utiliza una instrucción "WAIT" para describir un biestable D activado por flancos de subida.

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN ALGORÍTMICA

Ejemplo de utilización de la instrucción "WAIT UNTIL"

```
-- Descripción algorítmica en VHDL mediante
-- la instrucción "WAIT-UNTIL" de un biestable D
-- activado por flancos de subida representado
-- en la figura A1.48 del apéndice A1.
-- por Yago Mandado Junio 2007
-----------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-----------------------------------------------
-- Definición de la entidad biestable D
ENTITY biestable_D_por_flancos IS
 PORT (D, C          : IN std_logic;
          Q          : OUT std_logic);
 END biestable_D_por_flancos;
 -----------------------------------------------
-- Definición de la arquitectura algorítmica
-- (de comportamiento) de un biestable D
-- activado por flancos de subida, mediante
-- la instrucción "WAIT-UNTIL"
ARCHITECTURE algoritmo OF biestable_D_por_flancos IS
BEGIN
PROCESS
BEGIN
   WAIT UNTIL (C = '1');
      Q <= D;
END PROCESS;
END algoritmo;
```

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN ALGORÍTMICA

Instrucciones secuenciales: Instrucción "IF-THEN-ELSE"

Si se cumple (IF) una determinada condición se ejecutan (THEN) una o mas acciones (como por ejemplo asignaciones) secuenciales y en caso contrario (ELSE), otras diferentes.

Esta instrucción facilita la toma de decisión en función del nivel lógico de una o mas señales digitales. En el caso más sencillo esta constituida por una sentencia IF y otra ELSE, tal como se muestra en un ejemplo a continuación.

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN ALGORÍTMICA

Ejemplo de utilización de la instrucción "IF-THEN-ELSE"

```
-- Descripción algorítmica en VHDL del comportamiento
-- de un multiplexor de dos canales de entrada
-- por Yago Mandado Junio 2007
 ----------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
 ----------------------------------------------------------------------
-- Definición de la entidad de un multiplexor de dos canales
ENTITY multiplexor_dos_canales IS
PORT (D0, D1, S          : IN std_logic;
      Q                  : OUT std_logic);
END multiplexor_dos_canales;
 ----------------------------------------------------------------------
ARCHITECTURE algoritmo OF multiplexor_dos_canales IS
BEGIN
PROCESS (D0, D1, S)
BEGIN
--Compara la salida Q con la tabla de verdad
-- del multiplexor de dos canales
  IF (S = '0') THEN
       Q <= D1;
  ELSE
       Q <= D0;
  END IF;
END PROCESS;
END algoritmo;
```

DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN ALGORÍTMICA

Instrucciones secuenciales: Instrucción "CASE-WHEN"

Se utilizan para seleccionar una entre varias alternativas de ejecución de determinadas acciones en función del valor de una expresión. Su sintaxis se indica en la tabla.

```
[ etiqueta : ]
CASE expresión IS
    WHEN caso_1 => sentencias secuenciales
                caso_2 => sentencias secuenciales
…
END CASE [etiqueta];
```

LENGUAJE VHDL

DESCRIPCIÓN ALGORÍTMICA

Ejemplo de utilización de la instrucción "CASE-WHEN"

```
-- Descripción algorítmica en VHDL, con "CASE-WHEN",
-- de un multiplexor de cuatro canales
-- sin entrada de inhibición como el representado
-- con el símbolo lógico de la figura 3.92b del capítulo 3
 -- por Yago Mandado Junio 2007
 ----------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
 ----------------------------------------------------------------------------
ENTITY multiplexor4canales IS

PORT (D3, D2, D1, D0: IN std_logic;
                      S: IN bit_vector (1 DOWNTO 0);
                      Q: OUT std_logic);
 END multiplexor4canales;
----------------------------------------------------------------------------
-- Descripción algorítmica mediante la instrucción "CASE-WHEN"
ARCHITECTURE algoritmo OF multiplexor4canales IS
 BEGIN
    PROCESS(D3, D2, D1, D0, S)
    BEGIN
      CASE S IS
            WHEN "00" =>      Q <= D0;
            WHEN "01" =>      Q <= D1;
            WHEN "10" =>      Q <= D2;
            WHEN "11" =>      Q <= D3;
       END CASE;
    END PROCESS;
END algoritmo;
```

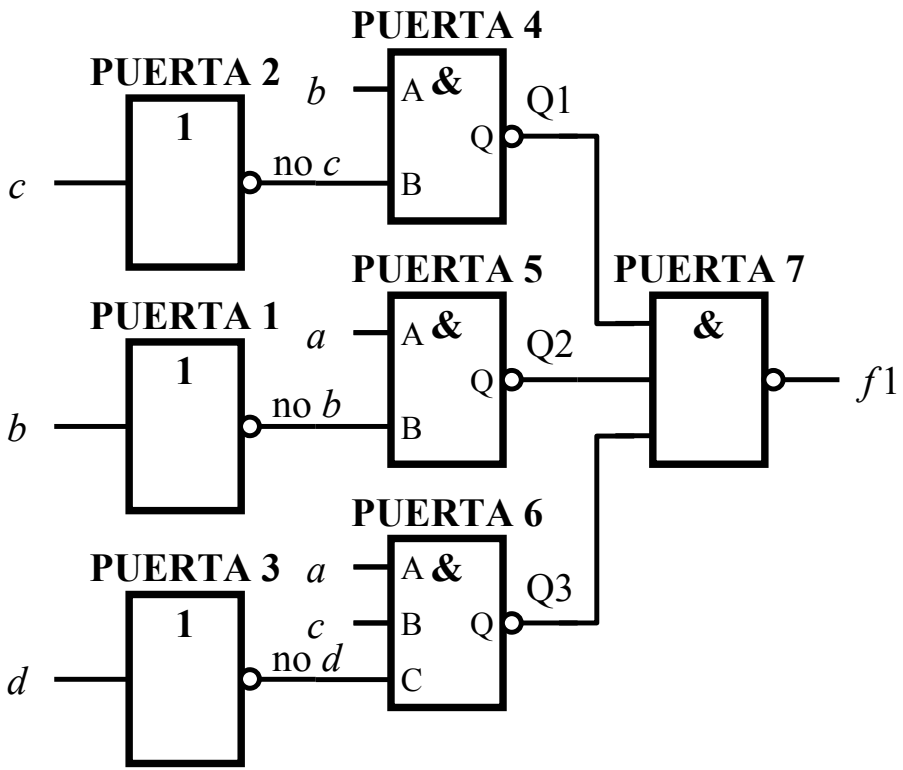
DESCRIPCIÓN DE UN SISTEMA DIGITAL EN VHDL

DESCRIPCIÓN ESTRUCTURAL

Se caracteriza por ser concurrente y no presenta las incertidumbres que tiene la descripción algorítmica, que, como en el caso de un multiplexor puede corresponder a tres circuitos reales diferentes. Es la más compleja y la que exige un conocimiento preciso de las características del sistema físico que se quiere describir.

La posibilidad de reutilizar (*Instantiate*) las descripciones estructurales hace que sean las más adecuadas cuando se quiere obtener un sistema de las mejores prestaciones posibles tanto en un circuito integrado a medida (ASIC) como mediante la configuración de una FPGA. La reutilización (*Instantiation*) permite llevar a cabo descripciones estructurales complejas a partir de otras más sencillas.

EJEMPLO DE DESCRIPCIÓN ESTRUCTURAL




```
-- Descripción estructural en VHDL del circuito combinacional
-- que implementa con puertas NAND la función f1
-- del ejemplo 3.2 (Figura 3.14) del capítulo 3
-- por Yago Mandado Junio 2007
 ------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
  ------------------------------------------------
-- Definición de la entidad función f1
ENTITY funcion f1 IS
 PORT (a, b, c, d) : IN std_logic;
                              f1: OUT std_logic);
 END funcion f1;
  ------------------------------------------------
 ARCHITECTURE estructural OF funcion f1 IS

COMPONENT NAND2entradas IS
PORT (x, y: IN std_logic; f: OUT std_logic);
END COMPONENT;

COMPONENT NAND3Entradas IS
PORT (x, y, z: IN std_logic; f: OUT std_logic);
END COMPONENT;

COMPONENT INV PORT (x: IN std_logic; f: OUT std_logic);
END COMPONENT;
-- Señales internas (Figura 9.10)
SIGNAL nob, noc, nod, Q1, Q2, Q3: std_logic;
BEGIN
-- Definición de las conexiones entre las puertas
puerta1: INV PORT MAP (b, nob);
puerta2: INV PORT MAP (c, noc);
puerta3: INV PORT MAP (d, nod);
puerta4: NAND2entradas PORT MAP (b, noc, Q1);
puerta5: NAND2entradas PORT MAP (a, nob, Q2);
puerta6: NAND3entradas PORT MAP (a, c, nod, Q3);
puerta7: NAND3entradas PORT MAP (Q1, Q2, Q3, f1);
END estructural;
```

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Es la mejor manera de comprobar el funcionamiento correcto de un sistema digital descrito en VHDL, antes de realizar su síntesis mediante alguna de las distintas formas de implementación posibles [Circuito integrado de aplicación específica (ASIC), PLD, FPGA, etc.].

Para realizar dicha comprobación, el lenguaje VHDL facilita la descripción de programas o bancos de prueba (*Test benches*).

La sintaxis de un programa de prueba depende en general de la herramienta utilizada pero debe, como mínimo, realizar las siguientes acciones:

- **Generar los estímulos o vectores de prueba.**

- **Aplicar o asignar los vectores a la entidad cuyo funcionamiento se verifica.**

Herramienta utilizada: Versión de estudiante de Modelsim

www.mentor.com

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Sintaxis del programa de prueba de la herramienta Modelsim

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
--------------------------------------------------------------------------------
ENTITY programa_prueba_[nombre];
END programa_prueba_[nombre];
--------------------------------------------------------------------------------
ARCHITECTURE [nombre] OF programa_prueba_[nombre] IS
COMPONENT [nombre del circuito a verificar] IS
PORT ([lista de señales de entrada y salida y su tipo]);
END COMPONENT;
SIGNAL [lista de señales de entrada y su tipo]
BEGIN
        [asignación de las señales de prueba al componente
         mediante PORTMAP]

 PROCESS
    BEGIN

      [generación de los vectores de prueba]

  END PROCESS;
END [nombre];
```

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

GENERACIÓN DE LOS ESTÍMULOS DE PRUEBA

La generación de los estímulos de prueba presenta dos aspectos:

- La definición de las señales que hay que generar para verificar el correcto funcionamiento.

- La forma en la que se programan las señales para que se realice la generación de las mismas.

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

GENERACIÓN DE LOS ESTÍMULOS DE PRUEBA

Definición de las señales de prueba

Para definir las señales de prueba que hay que aplicar a un circuito digital para verificar su correcto funcionamiento es necesario conocer la función que realiza. Cuando el circuito es sencillo, como los que se describen en los ejemplos de este capítulo, es fundamental que el lector conozca con precisión el comportamiento que tiene el circuito real para definir a partir de él las señales que hay que aplicarle.

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

GENERACIÓN DE LOS ESTÍMULOS DE PRUEBA

Programación de las señales de prueba

- Generación de vectores de prueba con “WAIT FOR”

- Generación de vectores de prueba con “AFTER”

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Sintaxis de la generación de señales de prueba mediante la orden "WAIT FOR".

```
PROCESS
  BEGIN

  [señal 1] <= [valor]
        .
        .
  [señal n] <= [valor];
  WAIT FOR t0 ns;
  [señal 1] <= [valor];
        .
        .
  [señal n] <= [valor];
  WAIT FOR t1 ns;
        .
        .
  [señal 1] <= [valor];
        .
        .
  [señal n] <= [valor];
  WAIT FOR tk ns;

 WAIT;                -- Finalización de la prueba
END PROCESS;
```

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Proceso de generación de una onda cuadrada de 20ns de período [generador de impulsos (*Clock*)] mediante la instrucción "WAIT FOR"

```
PROCESS
        BEGIN
            WAIT FOR 10 ns;
            C <= '1';
            WAIT FOR 10 ns;
            C <= '0';
 END PROCESS;
```

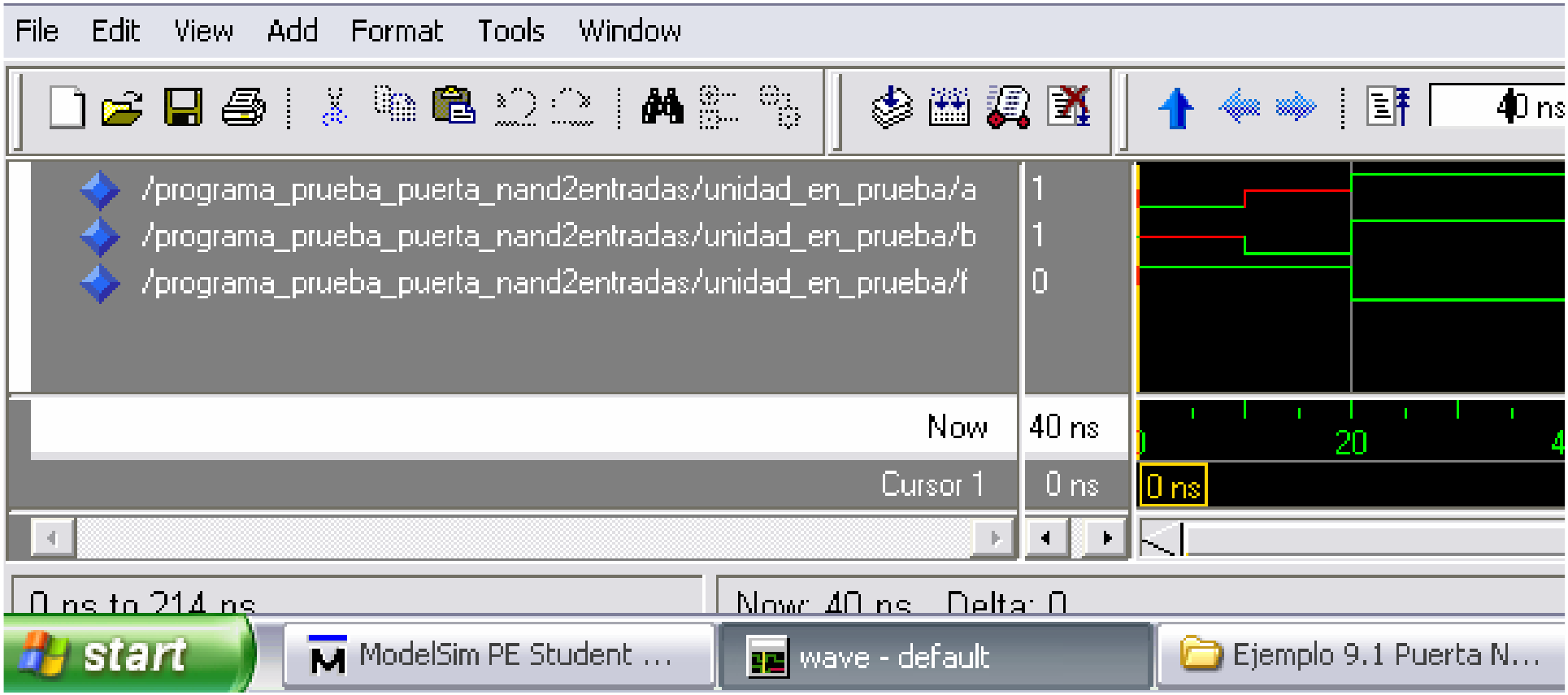
SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Ejemplo de programa de prueba que utiliza la instrucción WAIT FOR

```
-- Programa de prueba de la descripción de la puerta
-- NAND de dos entradas
-- por Yago Mandado Junio 2007
--------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
--------------------------------------------------------------------------------
-- Entidad vacia
ENTITY programa_prueba_puerta_NAND2entradas IS
END programa_prueba_puerta_NAND2entradas;
--------------------------------------------------------------------------------
ARCHITECTURE descripcion OF programa_prueba_puerta_NAND2entradas IS
COMPONENT puerta_NAND2entradas IS
PORT (a, b: IN std_logic;
          f: OUT std_logic);
END COMPONENT;
SIGNAL a, b: std_logic;
BEGIN

  Unidad_en_prueba: puerta_NAND2entradas PORT MAP (a, b);

 PROCESS
      BEGIN
      a <= '0';
      b <= 'X';
     WAIT FOR 10 ns;
     a <= 'X';
     b <= '0';
     WAIT FOR 10 ns;
      a <= '1';
      b <= '1';
      WAIT FOR 10 ns;

      WAIT;                                                    -- Finalización de la prueba

 END PROCESS;
END descripcion;
```

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Cronograma generado por el programa de prueba de la descripción de la puerta NAND de dos entradas



SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

GENERACIÓN DE LOS ESTÍMULOS DE PRUEBA

Generación de vectores de prueba con "AFTER"

Mediante la instrucción AFTER (Después) se pueden generar también tanto señales periódicas como no periódicas.

Una onda cuadrada simétrica (en la que la señal está tanto tiempo en nivel cero como uno) se puede generar, con algunos simuladores, mediante la asignación:

C <= **NOT** C **AFTER** t ns;

en la cual t es el valor del semiperíodo.

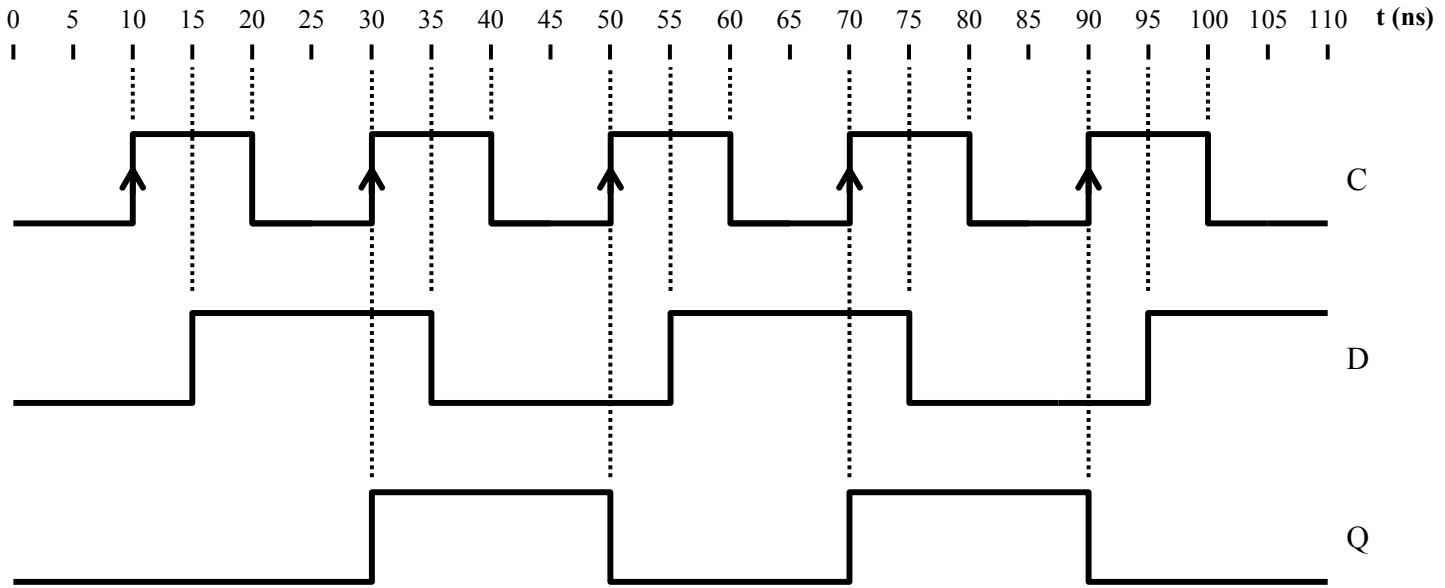
Una señal no periódica se genera mediante la asignación:

C <= '0', '1' **AFTER** t1 ns, '0'**AFTER** t2 ns, '1' **AFTER** t3 ns,.....;

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

EJEMPLO DE UTILIZACIÓN DE LA INSTRUCCIÓN AFTER

La elección de los instantes en los que deben cambiar las señales se realiza en función de las señales que se quieren aplicar al circuito que se prueba para verificar su correcto comportamiento. Por ejemplo, en el caso del biestable D activado por flancos se debe comprobar que la salida Q es igual al valor que tiene la entrada D cada vez que se aplica a su entrada C un flanco de subida.



SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Ejemplo de programa de prueba que utiliza la instrucción AFTER

```
-- Programa de prueba de la descripción en VHDL de
-- un biestable D activado por flancos de subida
-- como el representado en la figura A1.48 del apéndice A1
-- por Yago Mandado Junio 2007
-------------------------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-------------------------------------------------------------------------------------------------
ENTITY prueba_biestable_D_activado_por_flancos IS      -- Entidad vacia
END prueba_biestable_D_activado_por_flancos;
-------------------------------------------------------------------------------------------------
ARCHITECTURE descripcion OF prueba_biestable_D_activado_por_flancos IS
COMPONENT biestable_D_activado_por_flancos IS
PORT (D, C: IN std_logic;
                Q : OUT std_logic);
END COMPONENT;
SIGNAL D, C: std_logic;
BEGIN

  Unidad_en_prueba: biestable_D_activado_por_flancos PORT MAP (D, C);

   PROCESS
     BEGIN

C <= '0', '1' AFTER 10 ns, '0' AFTER 20 ns, '1' AFTER 30 ns, '0' AFTER 40 ns,
 '1' AFTER 50 ns, '0' AFTER 60 ns, '1' AFTER 70 ns, '0' AFTER 80 ns,
 '1' AFTER 90 ns, '0' AFTER 100 ns;
D <= '0', '1' AFTER 15 ns, '0' AFTER 35 ns, '1' AFTER 55 ns, '0' AFTER 75 ns;
WAIT;                    -- Finalización de la prueba
   END PROCESS;
 END descripcion;
```

SIMULACIÓN DE LA DESCRIPCIÓN EN VHDL DE CIRCUITOS Y SISTEMAS DIGITALES

Cronograma generado por el programa de prueba de la descripción del biestable D activado por flancos