Diseño Digital 1

Bloque temático 1

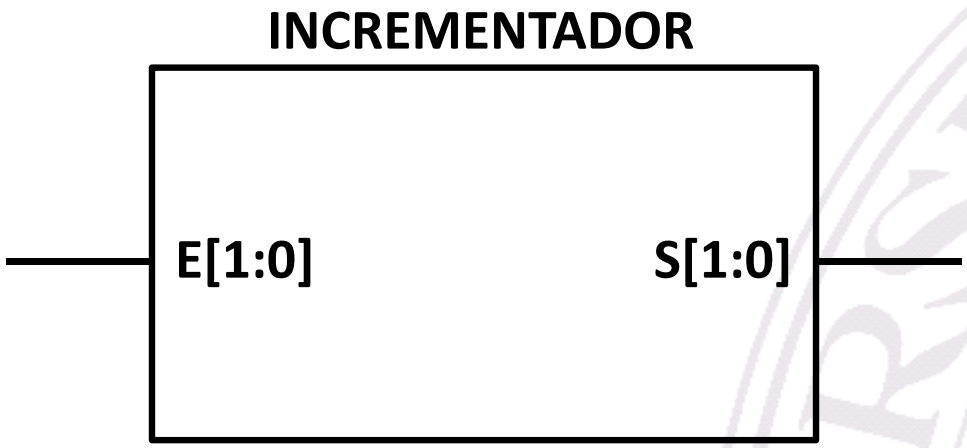
BT1.P1

Introducción a los HDLs

Modelado de Sistemas Combinacionales

Introducción a los HDLs

La descripción del modelo lógico de un circuito digital consiste en la definición de su Interfaz (características de sus entradas y salidas) y de su Funcionamiento (relación funcional entre entradas y salidas).



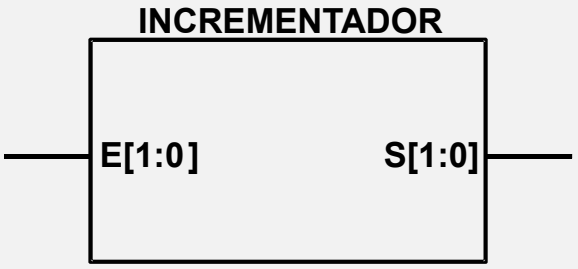
E1	E0	S1	S0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

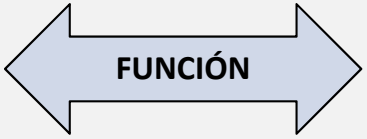
Los Lenguajes para la Descripción de Hardware (HDLs, *Hardware Description Languages*) permiten definir el modelo lógico de los sistemas digitales empleando un lenguaje formal.

Introducción a los HDLs

Los HDLs tienen una sintaxis similar a los lenguajes de programación, pero una semántica diferente: sirven para construir modelos lógicos de sistemas digitales.

```
library ieee;
use ieee.std_logic_1164.all;

entity INCREMENTADOR is
port(
     E: in     std_logic_vector(1 downto 0);
     S: buffer std_logic_vector(1 downto 0)
     );
end entity;

architecture RTL of INCREMENTADOR is
begin
  process(E)
  begin
    if  E = "00" then
      S <= "01";

    elsif E = "01" then
      S <= "10";

    elsif E = "10" then
      S <= "11";

    else
      S <= "00";
    end if;
  end process;
end RTL;
```



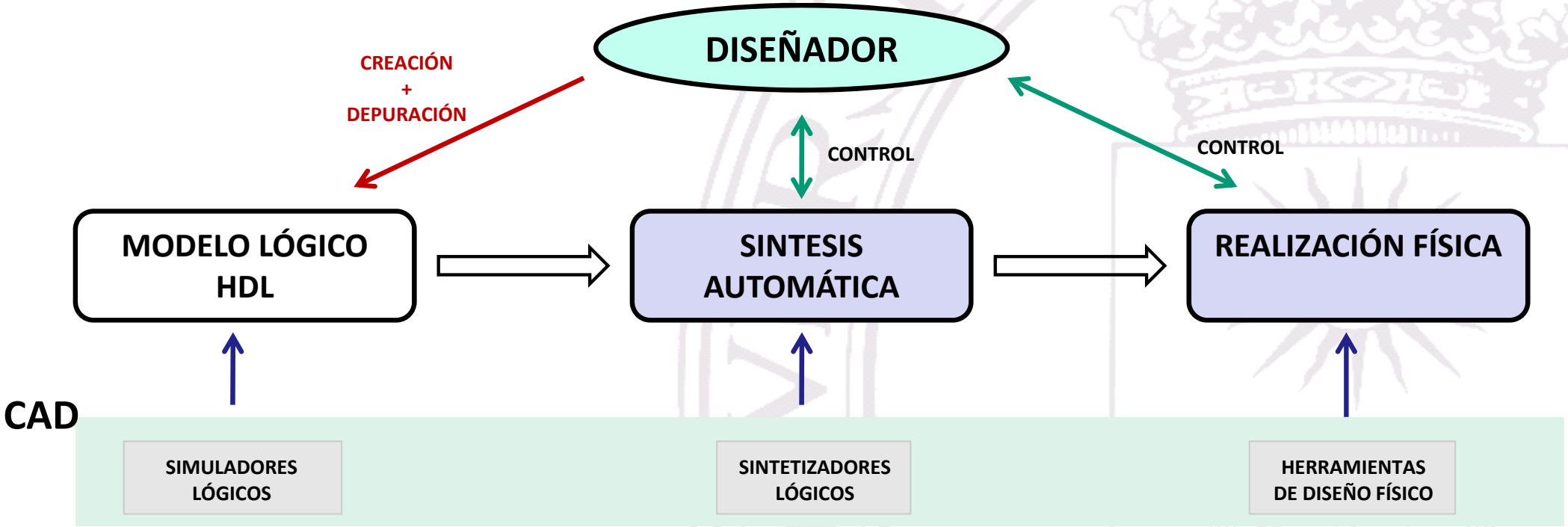


E1	E0	S1	S0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Introducción a los HDLs

Los HDLs son herramientas imprescindibles en las metodologías modernas de diseño de sistemas digitales VLSI, porque:

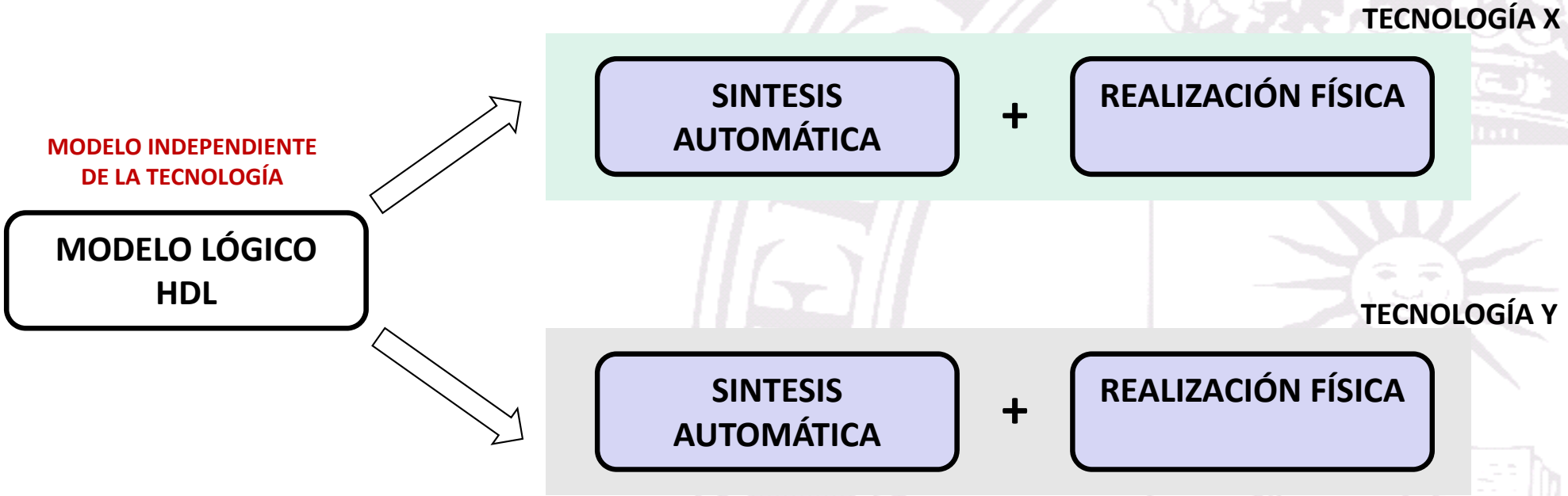
- Aumentan enormemente la eficiencia de las tareas de diseño
  - El diseñador crea el modelo lógico de los sistemas y utiliza herramientas de CAD para automatizar su síntesis lógica y su realización tecnológica.



Introducción a los HDLs

El uso de los HDLs proporciona muchas otras ventajas adicionales. Entre ellas destaca que:

- Permiten realizar diseños independientes de la tecnología de realización
  - No resulta necesario rehacer el modelo de un sistema digital para cambiar la tecnología con la que se materializa.



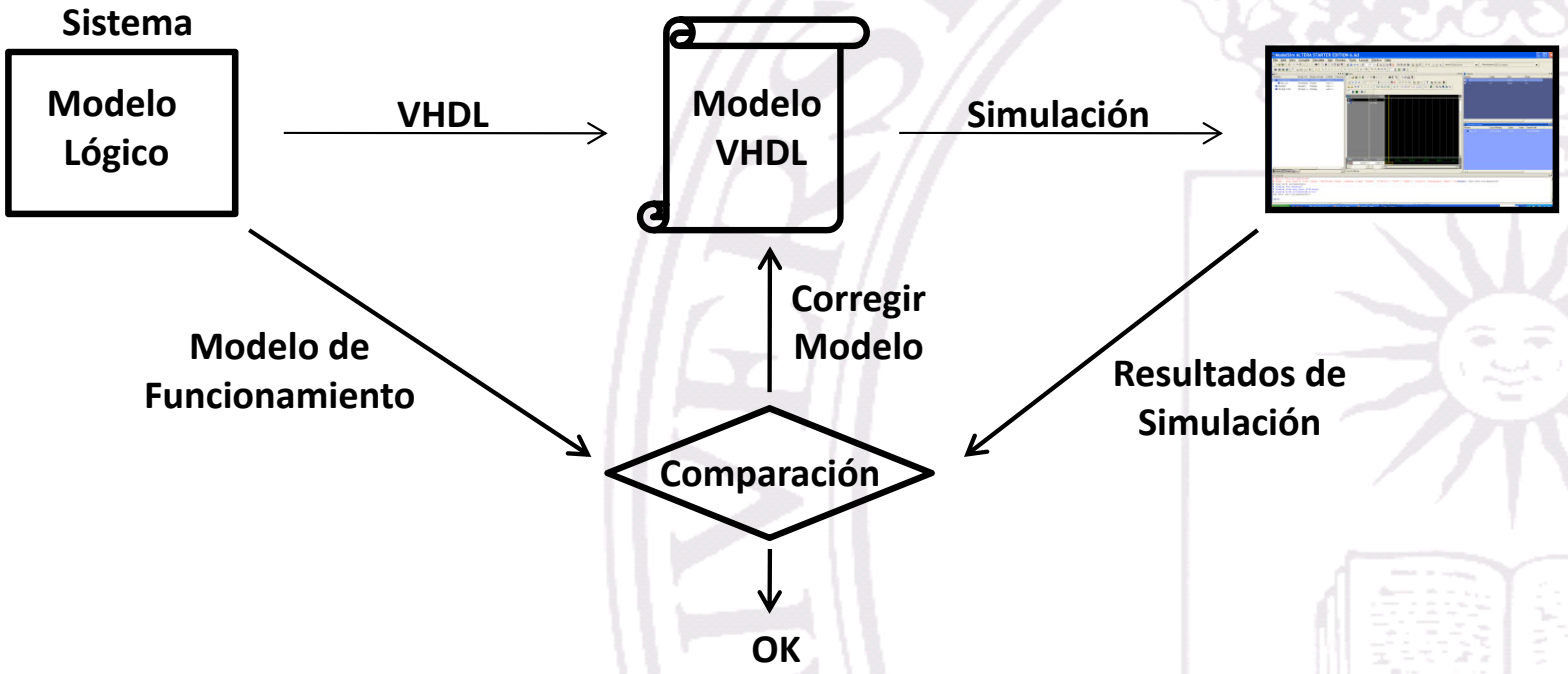
Enfoque del aprendizaje

- El Lenguaje de Descripción del Hardware que vamos a utilizar para construir modelos lógicos de sistemas digitales es el lenguaje VHDL: *VHSIC (Very High Speed Integrated Circuits) Hardware Description Language.*
    - Es un lenguaje normalizado por IEEE.
- En este curso adquirirá un conocimiento incompleto del lenguaje, pero que resultará suficiente para modelar y simular circuitos simples y de baja complejidad.
    - En Diseño Digital 2 aprenderá a modelar y simular circuitos más complejos con casi los mismos conocimientos del lenguaje.

Principios de modelado con VHDL

• El código de los modelos lógicos VHDL es ejecutable en un simulador VHDL: los modelos VHDL son modelos simulables.

• El modelo VHDL de un sistema digital es correcto si al simularlo su comportamiento coincide con el del sistema que se pretende modelar.

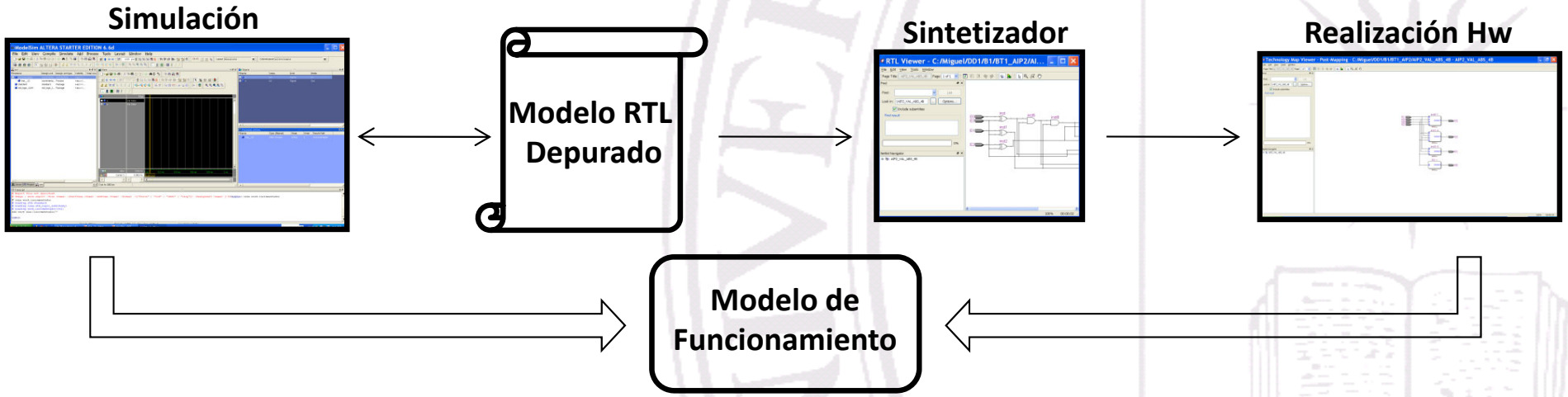


Principios de modelado con VHDL

• La principal ventaja del uso de modelos HDL es que pueden ser sintetizados automáticamente.

• Para que un modelo VHDL pueda ser sintetizado deben seguirse ciertas reglas en la codificación del modelo lógico. A los modelos que cumplen estas reglas se los suele denominar modelos RTL.

• Una herramienta de síntesis es capaz de generar, a partir de modelos RTL, circuitos que funcionan igual que los modelos en una simulación.



Modelado RTL con VHDL

• En VHDL la interfaz de los sistemas digitales y su funcionamiento se describen separadamente, en dos unidades de código distintas: la interfaz en una Declaración de Entidad y el funcionamiento en un Cuerpo de Arquitectura.

```
LIBRARY ieee:
USE ieee.std_logic_1164.all;

ENTITY selector8bits IS
PORT(DatoA: IN std_logic_vector(7 downto 0);
        DatoB: IN std_logic_vector(7 downto 0);
        Sel:     IN std_logic;
        DatoSel: BUFFER std_logic_vector(7 downto 0));
END ENTITY;

ARCHITECTURE rtl OF selector8bits IS
BEGIN
      PROCESS(DatoA, DatoB, Sel)
      BEGIN
         IF Sel = '0' THEN
           DatoSel <= Datoa;

        ELSE
          DatoSel <= DatoB;
        END IF;
      END PROCESS;
END rtl;
```

Modelado de la Interfaz

Modelado del Funcionamiento

Modelado de la Interfaz

- En la Declaración de Entidad se definen:
  - El nombre del circuito.
  - Los puertos (entradas y salidas) del circuito, indicando su nombre, dirección (de entrada, salida o bidireccional) y el tipo de datos asociado (que, entre otras cosas, permite indicar el número de bits que componen el puerto en cuestión).

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

ENTITY nombre_del_circuito IS
PORT(nombre_puerto : dirección tipo_de_dato;
     nombre_puerto : dirección tipo_de_dato;
     nombre_puerto : dirección tipo_de_dato);
END ENTITY;
```

Modelado de la Interfaz

Las etiquetas de usuario (el nombre de la entidad y los nombres de los puertos) deben elegirse de modo que se correspondan con la función que realizan.

- La dirección de los puertos se indica con las palabras:
  - IN para los puertos de entrada
  - BUFFER para los puestos de salida
  - INOUT para los bidireccionales

- Los tipos de datos que se utilizan son:
  - std_logic para los puertos de 1 bit
  - std_logic_vector para los puertos compuestos por varios bits

Modelado de la Interfaz

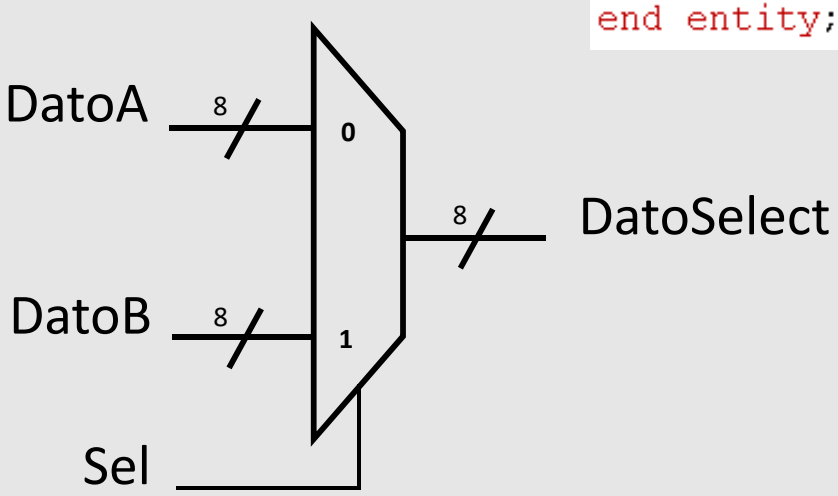
SELECTOR DE DATOS
DE 8 BITS

```
library ieee;
use ieee.std_logic_1164.all;

entity Selector8bits is
port(DatoA: in std_logic_vector(7 downto 0);
     DatoB: in std_logic_vector(7 downto 0);
     Sel:   in std_logic;
     DatoSelect: buffer std_logic_vector(7 downto 0)
   );
end entity;
```

(1)

(2)

(3)



(1) Librerías
(2) Nombre del modelo
(3) Lista de puertos (nombre, dirección y tipo)

Modelado de la Interfaz

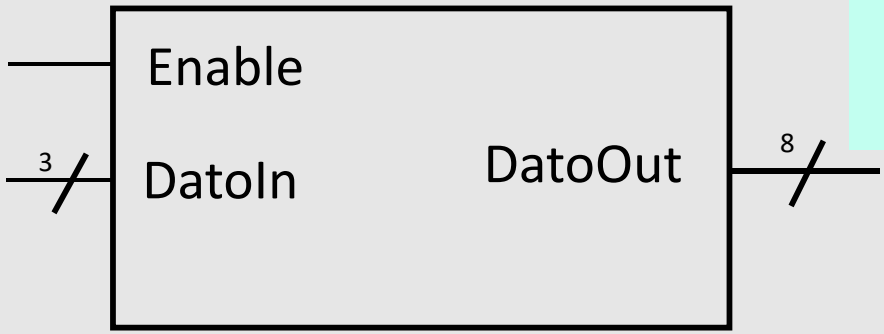
DECODIFICADOR DE 8 BITS

```
library ieee;
use ieee.std_logic_1164.all;

entity Decodificador3a8 is
port(Enable: in std_logic;
     DatoIn: in std_logic_vector(2 downto 0);
     DatoOut: buffer std_logic_vector(7 downto 0)
    );
end entity;
```

(1)

(2)

(3)

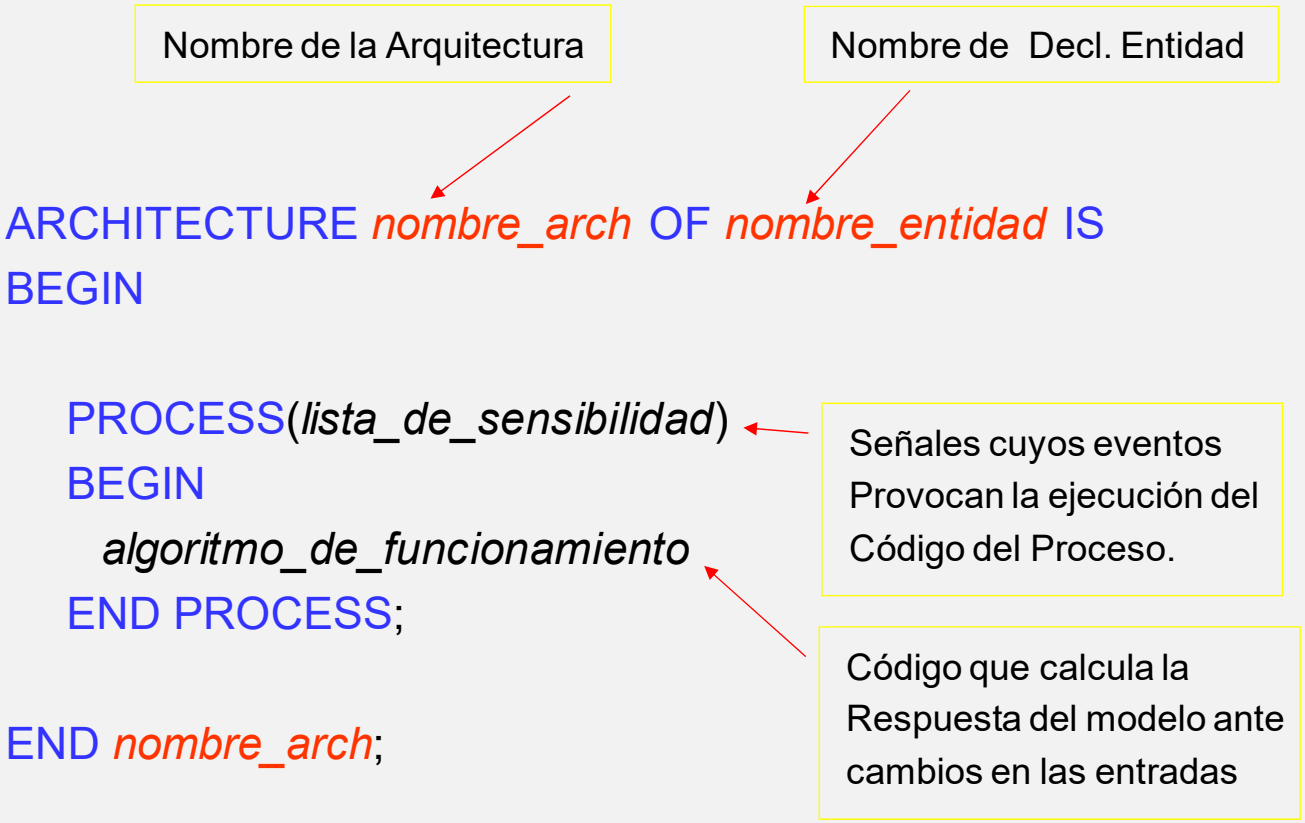


(1) Librerías
(2) Nombre del modelo
(3) Lista de puertos (nombre, dirección y tipo)

Modelado RTL de circuitos Combinacionales

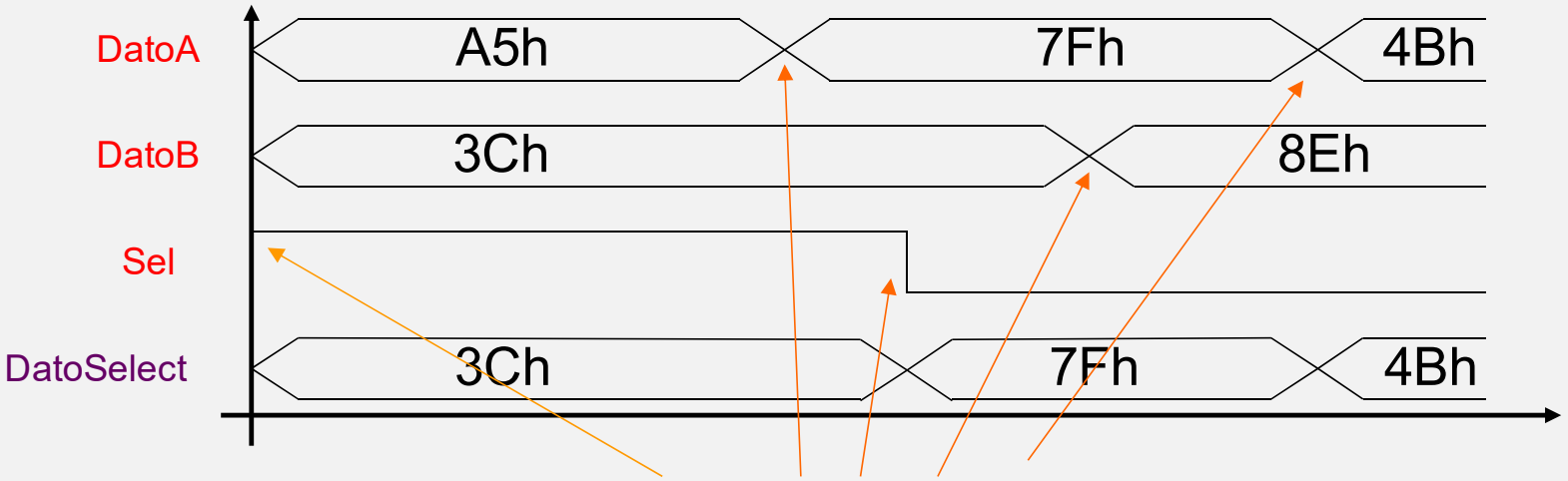
En VHDL el funcionamiento de los circuitos digitales se describe en una unidad de código denominada Cuerpo de Arquitectura.

Dentro de los Cuerpos de Arquitectura el funcionamiento de los circuitos simples se expresa utilizando unas construcciones VHDL denominadas Procesos.



Modelado RTL de circuitos Combinacionales

Un proceso contiene código de ejecución secuencial (como el de un programa realizado con un lenguaje de programación) que se ejecuta (durante una simulación), cuando hay cambios de valor (eventos, en jerga HDL) en alguna de las señales a las que es sensible el proceso.




```
PROCESS(DatoA, DatoB, Sel)
BEGIN
   IF Sel = '0' THEN
     DatoSel <= DatoA;
   ELSE
     DatoSel <= DatoB;
   END IF;
END PROCESS;
```

Ejecución en una simulación de un proceso que modela un selector de datos de 8 bits

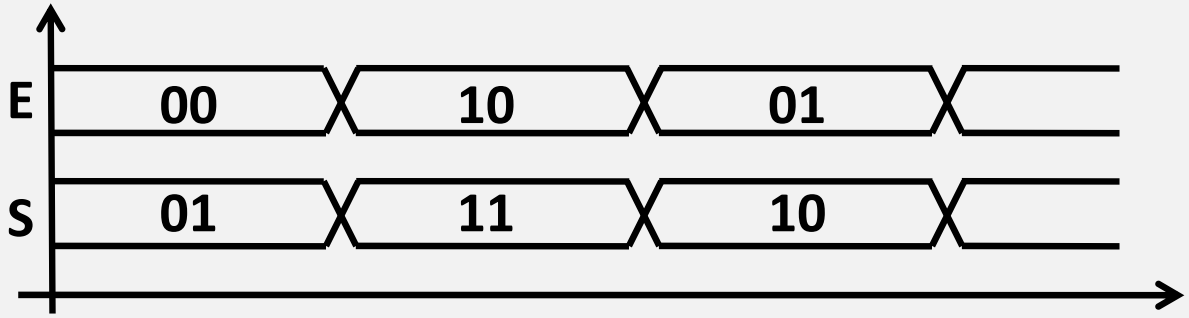
Modelado RTL de circuitos Combinacionales

En un circuito combinacional, el valor (la combinación binaria) de sus salidas depende exclusivamente, en cualquier instante, del valor (la combinación binaria) que simultáneamente haya en sus entradas.

El valor de los bits de salida sólo puede variar, por tanto, cuando haya un cambio en la combinación de bits de entrada.

Para definir el funcionamiento de un circuito combinacional hay que especificar, de manera exhaustiva, el valor que toma cada bit de salida para todas y cada una de las posibles combinaciones de los bits de entrada.

E1	E0	S1	S0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0



Modelado RTL de circuitos Combinacionales

Para codificar mediante un proceso el modelo sintetizable de un circuito combinacional hay que seguir las siguientes reglas:

- El proceso tiene que ser sensible a TODAS las entradas del circuito que modela.

- El código del proceso debe asignar valor a todos los bits de salida para todas las posibles combinaciones de los bits de entrada. Para ello resulta suficiente utilizar sentencias IF y CASE, operaciones aritméticas y lógicas y sentencias de asignación.

E1	E0	S1	S0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

```
process(E)
begin
  case  E  is
    when "00" => S <= "01";
    when "01" => S <= "10";
    when "10" => S <= "11";
    when "11" => S <= "00";
    when others => null;

  end case;
end process;
```

Sintaxis para el modelado de Circuitos Combinacionales

1. Lista de Sensibilidad: Es la lista de nombres de los puertos de entrada declarados en la Declaración de Entidad. Va a continuación de la palabra PROCESS:

   PROCESS(entrada1, entrada2, entrada3)

2. Sentencias de asignación: Indican el valor que toma un puerto de salida del circuito modelado:

   Salida <= *Valor asignado*;

   El valor asignado puede ser:

   - Un nivel lógico (si la salida es de un bit): S <= '0' ó S <= '1'
   - Un grupo de niveles lógicos (si la salida es un bus): p. ej. S <= "00"
   - Un puerto de entrada. S <= I
   - Un grupo de puertos de entrada entre los que se realiza una operación lógica o aritmética: S <= A + B ó S <= A OR B

3. Operaciones Lógicas: AND, OR, NAND, NOR, XOR

Sintaxis para el modelado de Circuitos Combinacionales

Sentencias IF:

- IF *cláusula_de_selección* THEN
  sentencias
  ELSE
  sentencias
  END IF;

- IF *cláusula_de_selección* THEN
  sentencias
  ELSIF *cláusula_de_selección* THEN
  sentencias
  ELSE
  sentencias
  END IF;

Sentencia CASE:

```
CASE vector_entrada IS
 WHEN valor =>
      sentencias
 WHEN valor1 | valor2 | … =>
      sentencias
 WHEN OTHERS =>
      sentencias
END CASE;
```

Modelado de Circuitos Combinacionales

E1	E0	S1	S0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

```
library ieee;
use ieee.std_logic_1164.all;

entity INCREMENTADOR is
port(
     E: in std_logic_vector(1 downto 0);
     S: buffer std_logic_vector(1 downto 0)
     );
end entity;

architecture RTL of INCREMENTADOR is
begin
  process(E)
  begin
    case(E) is
      when "00" => S <= "01";
      when "01" => S <= "10";
      when "10" => S <= "11";
      when "11" => S <= "00";
      when others => S <= "XX";
    end case;
  end process;
end RTL;
```