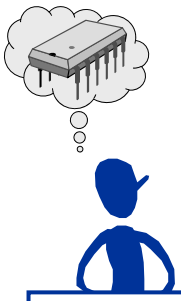
Simulación de circuitos descritos en VHDL

Autores:
Celia López
Luis Entrena
Mario García
Enrique San Millán
Marta Portela
Almudena Lindoso

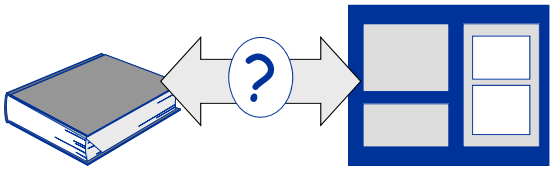
BY NC SA

Indice

Validación funcional

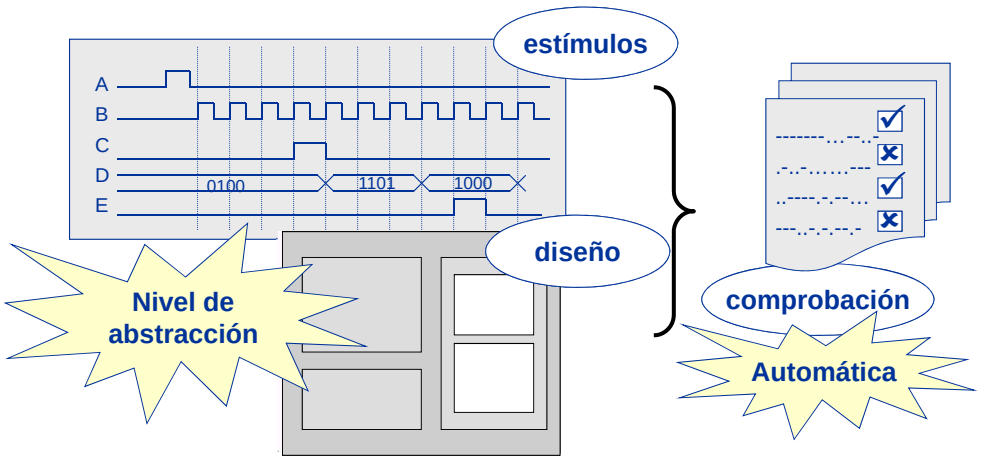


Comprobación de la funcionalidad del diseño de acuerdo a las especificaciones

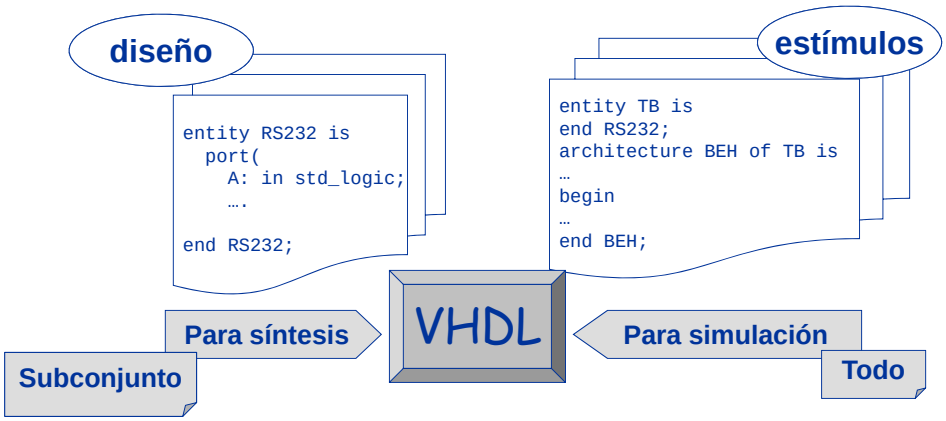


Una vez descrito el diseño ¿cómo podemos estar seguros de que funciona de acuerdo a las especificaciones?

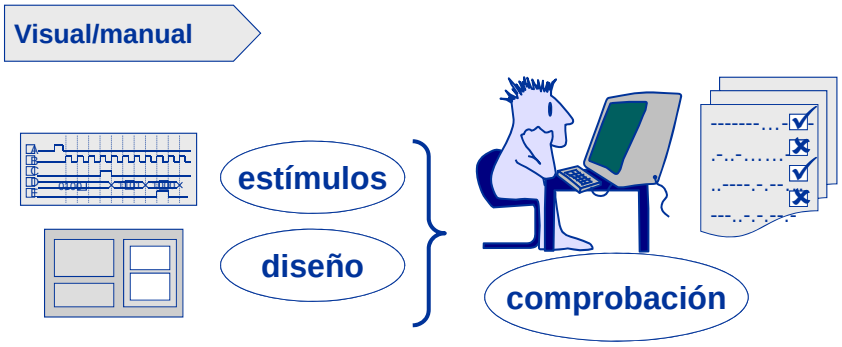
Validación funcional: Elementos necesarios



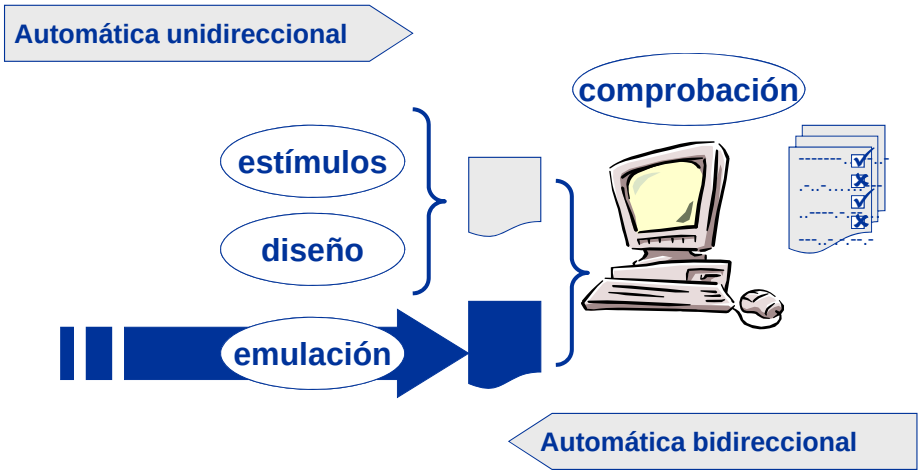
Validación funcional: Elementos necesarios



Niveles de validación funcional



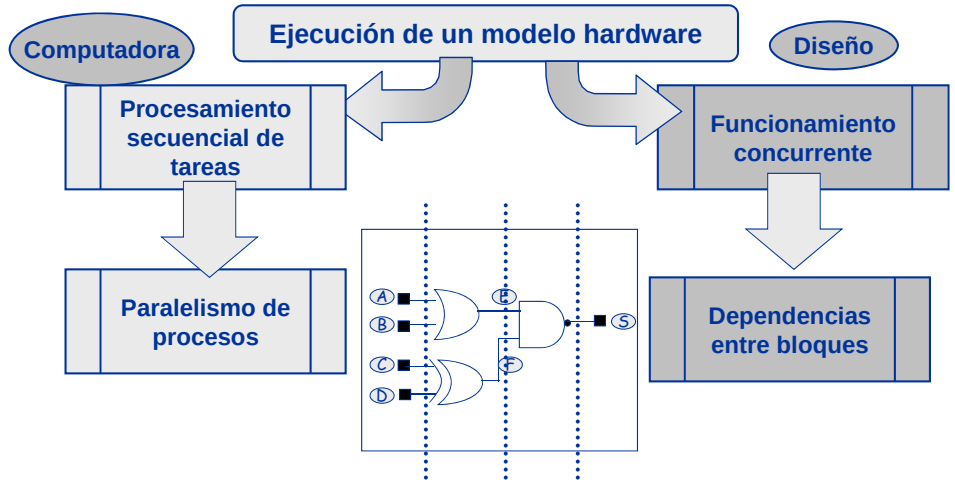
Niveles de validación funcional



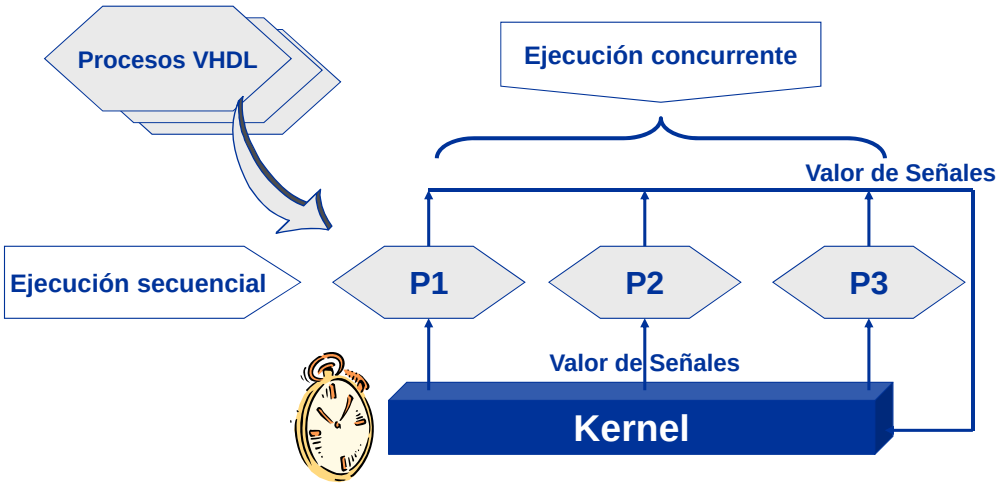
Validación funcional con VHDL

- Lenguaje concebido para simulación y especificación
- Posibilidad de comprobación automática e interactiva
- Modelado de alto nivel de interfaces externas
- Mecanismos de escritura/lectura de ficheros
- Especificaciones de tiempos y retardos

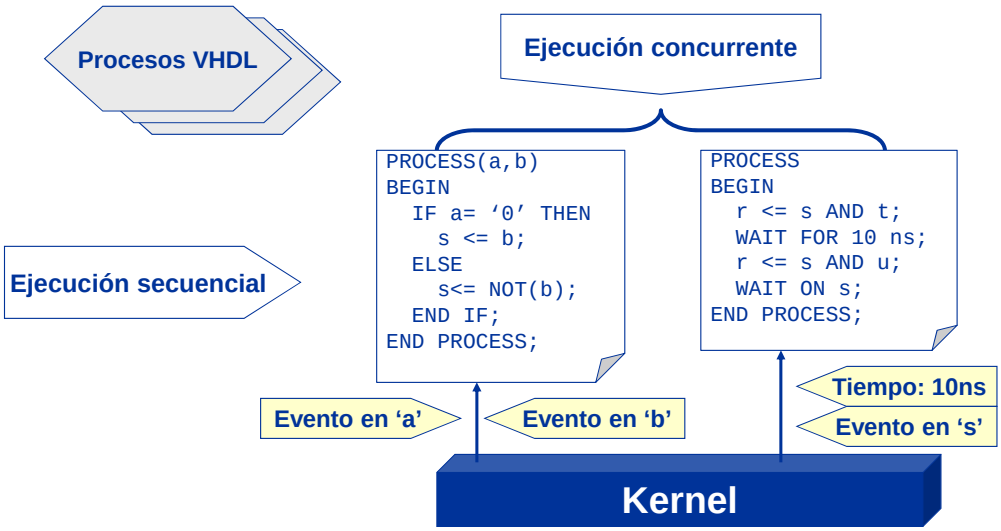
Simulación de circuitos digitales



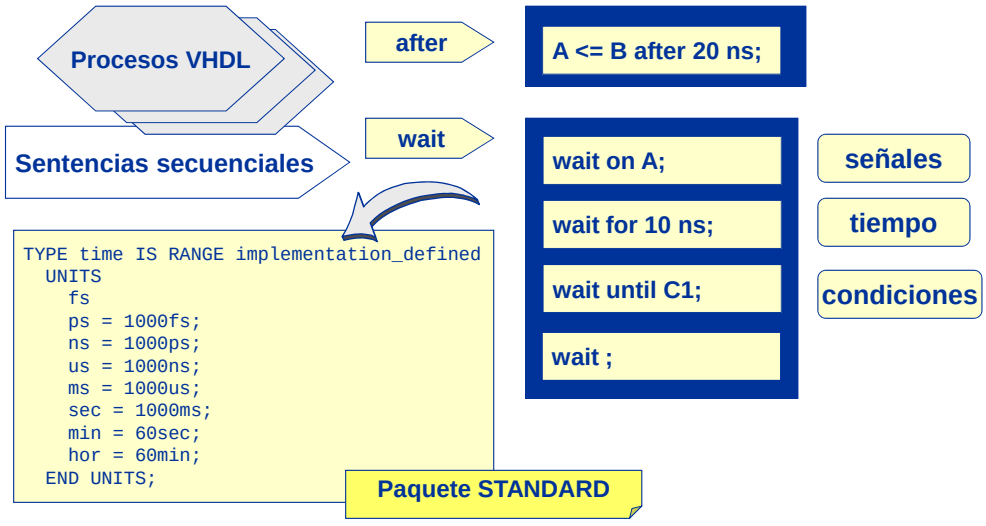
Simulación de circuitos digitales



Simulación de circuitos digitales



Simulación de circuitos digitales



Banco de pruebas

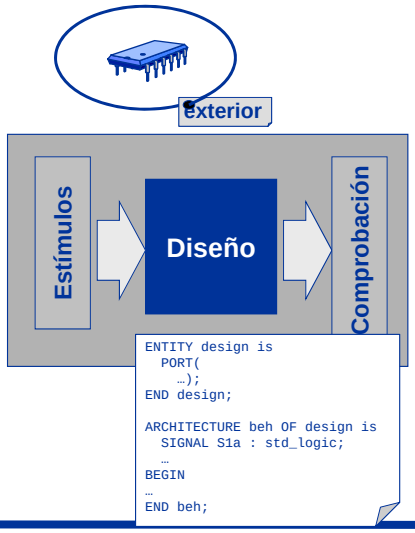
estímulos

- Lo más completo posible — ¿Todas las combinaciones?
- Mayor nivel de abstracción posible — Traducción automática
- Comprobación lo más exhaustiva posible — Evitar factores externos

Reglas básicas del Banco de pruebas

- Inicialización asíncrona de todo el sistema
- Todos los modos de operación / Estados de las FSMs
- Lectura y Escritura de todos los registros — Reflejo en las salidas
- Todos los modos de operación de los buses
- Cobertura de código

Partes básicas del Banco de pruebas




```
ENTITY design is
  PORT(
    …);
END design;

ARCHITECTURE beh OF design is
  SIGNAL S1a : std_logic;
  …
BEGIN
…
END beh;
```

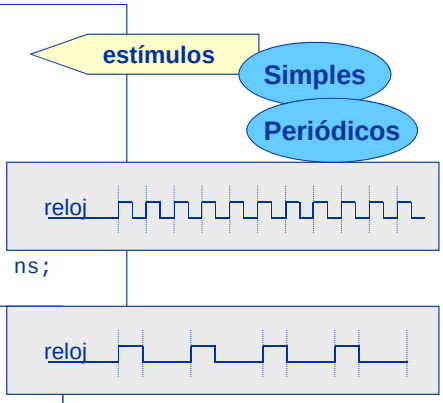
```
ENTITY tb is
END tb;
ARCHITECTURE mixed OF tb is
  COMPONENT design IS
    PORT(
      …);
  END COMPONENT;
  FOR D: design
    USE entity WORK.design(beh)
  SIGNAL s1 : std_logic;
  …
BEGIN
  D: design
  PORT MAP(
      …);
  …
  -- ESTIMULOS !!!
  -- COMPROBACIONES !!!
END mixed;
```

entidad vacía
componente
especificación
E/S y más
mapeo
estímulos
comprobaciones

Generación de estímulos: Relojes

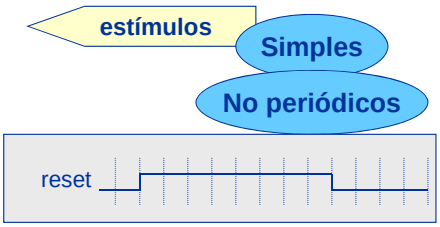
```
ENTITY tb is
END tb;

ARCHITECTURE mixed OF tb is
  SIGNAL reloj: std_logic := '0';
  CONSTANT semi_periodo : natural := 20;
BEGIN
…
  -- ESTIMULOS !!!
  reloj <= NOT(reloj) after semi_periodo ns;
END mixed;
```

```
PROCESS
BEGIN
  reloj <= '0';
  WAIT FOR semi_periodo_L ns;
  reloj <= '1';
  WAIT FOR semi_periodo_H ns;
END PROCESS;
```



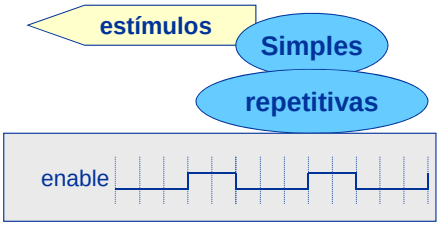
Generación de estímulos: Reset

```
ENTITY tb is
END tb;

ARCHITECTURE mixed OF tb is
  SIGNAL reset: std_logic;
  CONSTANT reset_on : natural := 1000;
BEGIN
…
  -- ESTIMULOS !!!
  PROCESS
  BEGIN
    reset <= ‘0’;
    WAIT FOR 150 ns;
    reset <= ‘1’;
    WAIT FOR reset_on ns;
    reset <= ‘0’;
    WAIT;
  END PROCESS;
END mixed;
```



Generación de estímulos: Enable

```
ENTITY tb is
END tb;

ARCHITECTURE mixed OF tb is
  SIGNAL enable: std_logic;
BEGIN
…
  -- ESTIMULOS !!!
  PROCESS
  BEGIN
    enable <= ‘0’;
    WAIT FOR 150 ns;
    enable <= ‘1’;
    WAIT FOR 100 ns;
    enable <= ‘0’;
  END PROCESS;
END mixed;
```



Generación de estímulos: Datos entrada

```
ENTITY tb is
END tb;

ARCHITECTURE mixed OF tb is
  SIGNAL count: integer range 0 to 3 := 0;
BEGIN
…
  -- ESTIMULOS !!!
  PROCESS
  BEGIN
    IF count = 3 THEN
      count = 0;
    ELSE
      count <= count + 1;
    END IF;
    WAIT FOR 150 ns;
  END PROCESS;
END mixed;
```

estímulos

Complejas

contadores

Generación de estímulos: Memorias

```
…
ARCHITECTURE mixed OF tb is
  TYPE TableType is array (natural range <>) of
                    std_logic_vector(3 downto 0);
  SIGNAL ValueTable: TableType (0 to 20)
                   := ("0011", "0110", "0111",
                       "1011", "1110", "1111",
                       "1001", "0111", "0101",
                       "1010", "0111", "0011",
                       "0000", "1110", "0110",
                       "0100", "0001", "0101",
                       "0010", "0100");
BEGIN
  PROCESS(Index)
  BEGIN
   Value <= ValueTable(Index);
END PROCESS;
END mixed;
```

estímulos

Complejas

tablas

Comprobación en el banco de pruebas

```
ENTITY tb is
END tb;

ARCHITECTURE mixed OF tb is
  -- Cuenta de 0 a 10
  SIGNAL s1 : std_logic_vector(3 downto 0);
  …
BEGIN
  -- COMPROBACIONES !!!
  -- Comprueba que s1 no ha superado su valor sup.
  ASSERT CONV_INTEGER(s1) < 11
    REPORT "Overflow in Count"
     SEVERITY warning;
  -- Comprueba que no violamos el tiempo de Hold
  ASSERT (Now – LastEvent) >= HoldTime
    REPORT "Hold time violation"
      SEVERITY warning;
END mixed;
```

comprobación

assert

condición

false

report

texto

severity

NOTE

WARNING

ERROR

FAILURE

Detienen la simulación

Paquete STANDARD

Interacción con el exterior

Ficheros de entrada y salida

- Un tipo FILE define un tipo de datos contenidos en un fichero
- Una declaración de FILE define un identificador para un fichero y lo asocia a un fichero físico
- Los objetos de tipo fichero no pueden recibir asignaciones, pero pueden ser leidos y escritos a través de subprogramas especiales:
  - Procedimiento READ (tipo_fichero, dato)
  - Procedimiento WRITE (tipo_fichero, dato)
  - Función ENDFILE (tipo_fichero)
- El paquete TEXTIO, disponible en todos los entornos VHDL, define tipos, procedimientos y funciones para lectura y escritura de ficheros ASCII

Ficheros E/S. Lectura de estímulos

```
ARCHITECTURE mixed OF tb is
-- Declaración de fichero. Tipo TEXT
  FILE input_file: TEXT IS IN "input.dat";
BEGIN
  -- ESTIMULOS !!!
  PROCESS
    VARIABLE val1 : char;
    VARIABLE val2 : integer;
    VARIABLE linea: LINE;
  BEGIN
    IF not(ENDFILE(input_file)) THEN
      -- Para leer una línea de texto
      READLINE (input_file, linea);
      -- El dato leído se interpreta como un char
      READ (linea, val1);
      -- El dato leído se interpreta como un integer
      READ (linea, val2);
    END IF;
    Dato_Input <= Val2;
    WAIT FOR 100 ns;
  END PROCESS;
END mixed;
```

estímulos

comprobación

E/S

Ficheros E/S. Escritura de resultados

```
ARCHITECTURE mixed OF tb is
  -- Declaración de fichero
  FILE output_file: TEXT IS OUT "output.dat";
BEGIN
  -- ESTIMULOS !!!
  PROCESS(Dato)
    VARIABLE linea: LINE;
  BEGIN
    -- Para escribir una línea
    WRITE (linea, Dato);
    WRITELINE (output_file, linea);
    WAIT FOR 100 ns;
  END PROCESS;
END mixed;
```

estímulos

comprobación

E/S

Herramientas comerciales