Tema 4:

Simulación con un banco de pruebas VHDL - *test bench*.

4.1 Introducción

4.2 Diseño de un *test bench*

4.3 Ejemplos

Tema 4:

Simulación con un banco de pruebas VHDL - *test bench*.

4.1 Introducción

4.2 Diseño de un *test bench*

4.3 Ejemplos

Introducción

- Simulación
  - Para simular un diseño es necesario generar estímulos para todas las entradas.
  - Algunos programas (versión anterior de ISE Xilinx) permiten generar las entradas de forma interactiva.
  - En la mayoría de los casos es necesario escribir una descripción del entorno conocido como un *test bench*.

Tema 4:

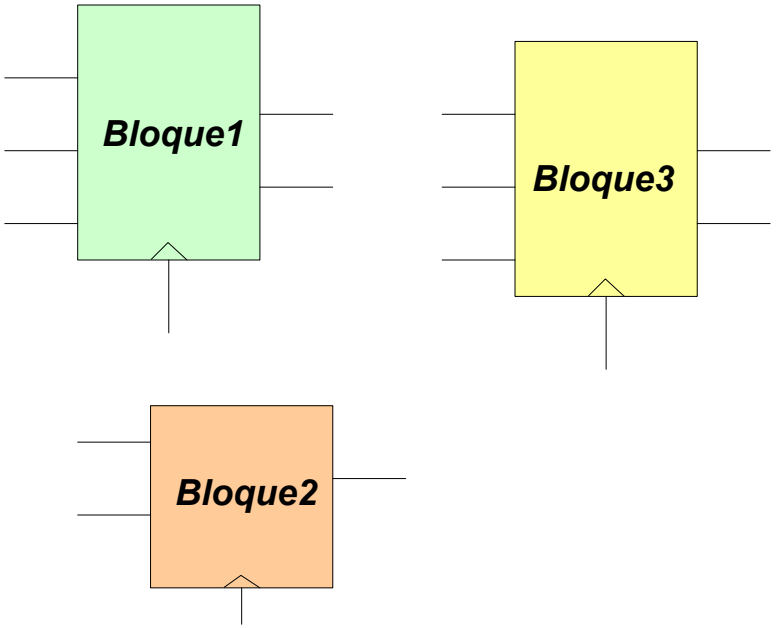
Simulación con un banco de pruebas VHDL - *test bench*.

4.1 Introducción

4.2 Diseño de un test bench

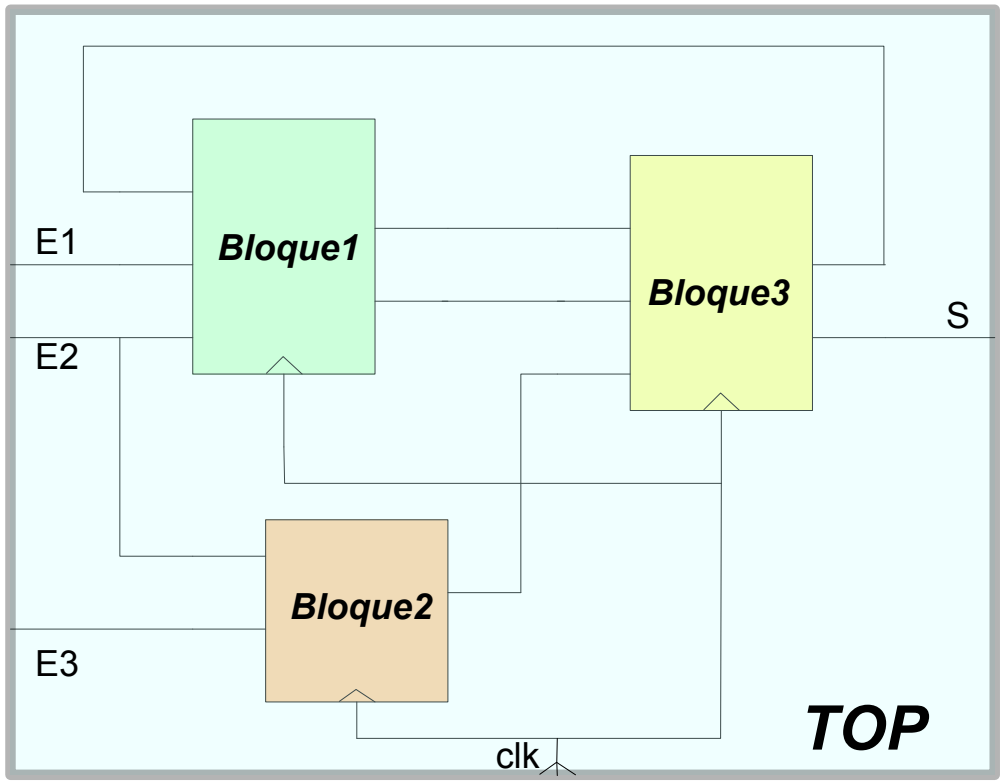
4.3 Ejemplos

Diseño de un *test bench*



En primer lugar se diseñan los bloques básicos del diseño

Diseño de un *test bench*

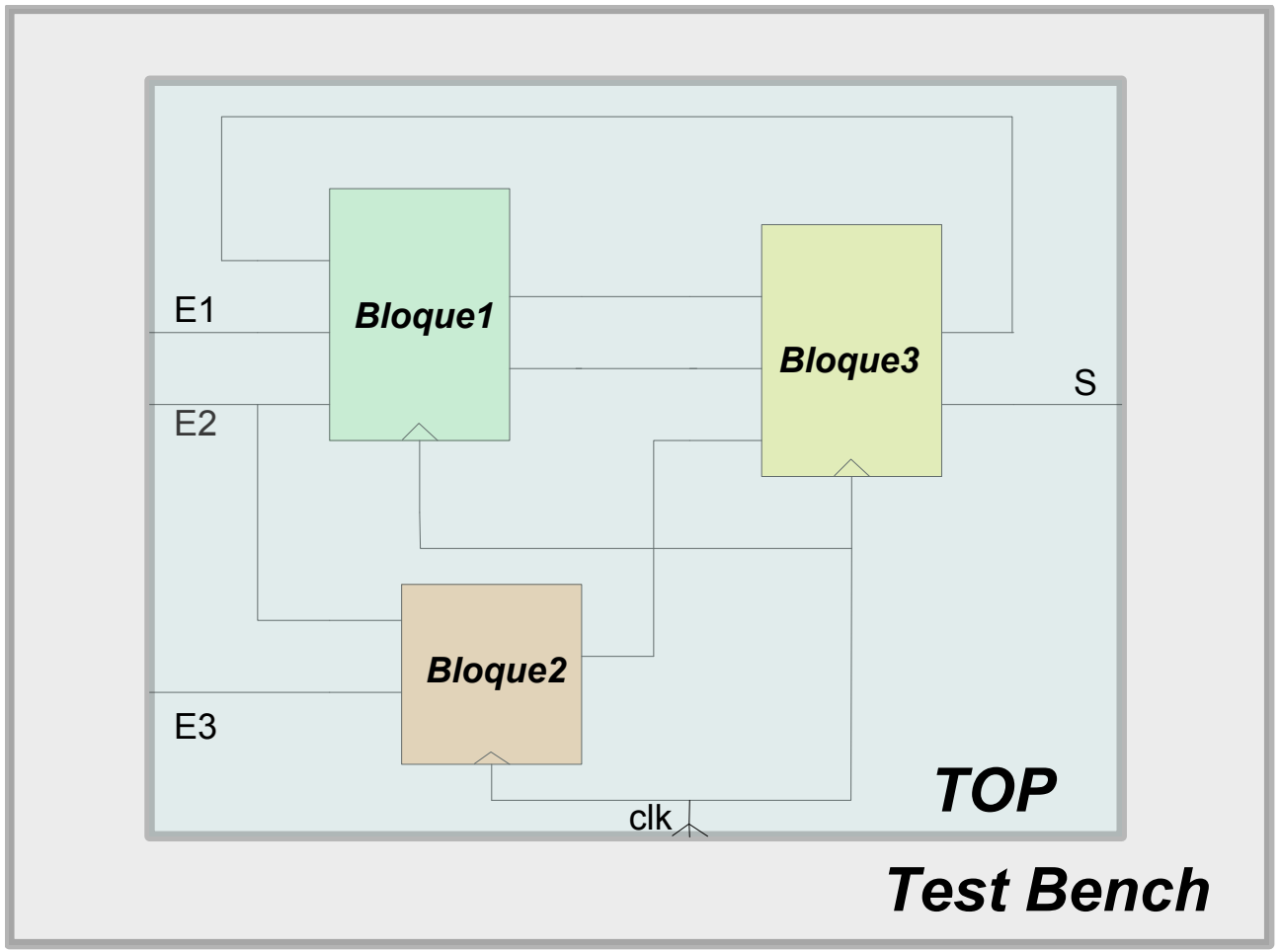


Se conectan los bloques para formar un bloque estructural

- Cables que no salen al exterior: signals

- Cables que salen al exterior: ports

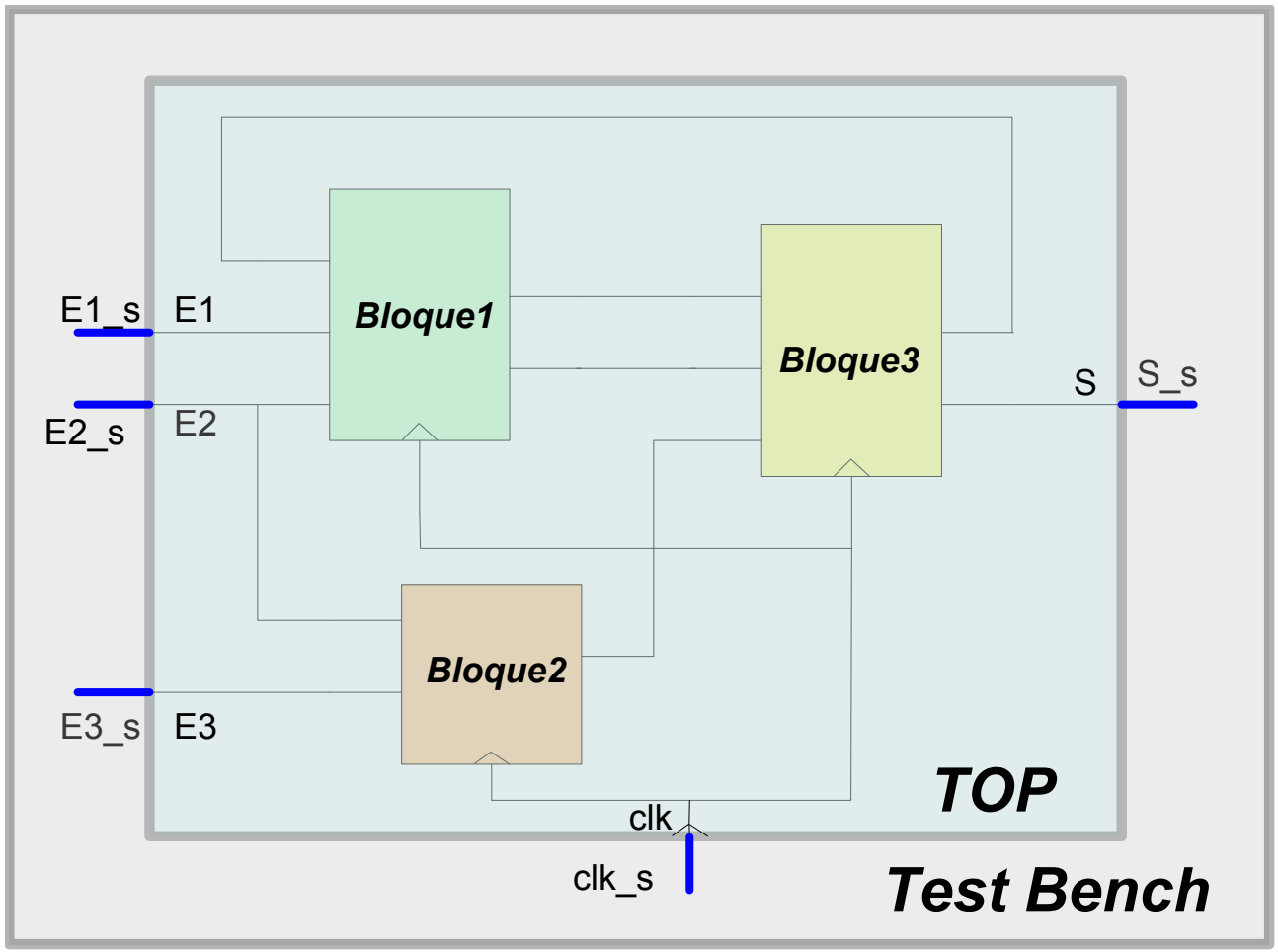
Diseño de un *test bench*



Se incluye nuestro diseño en un bloque superior sin puertos

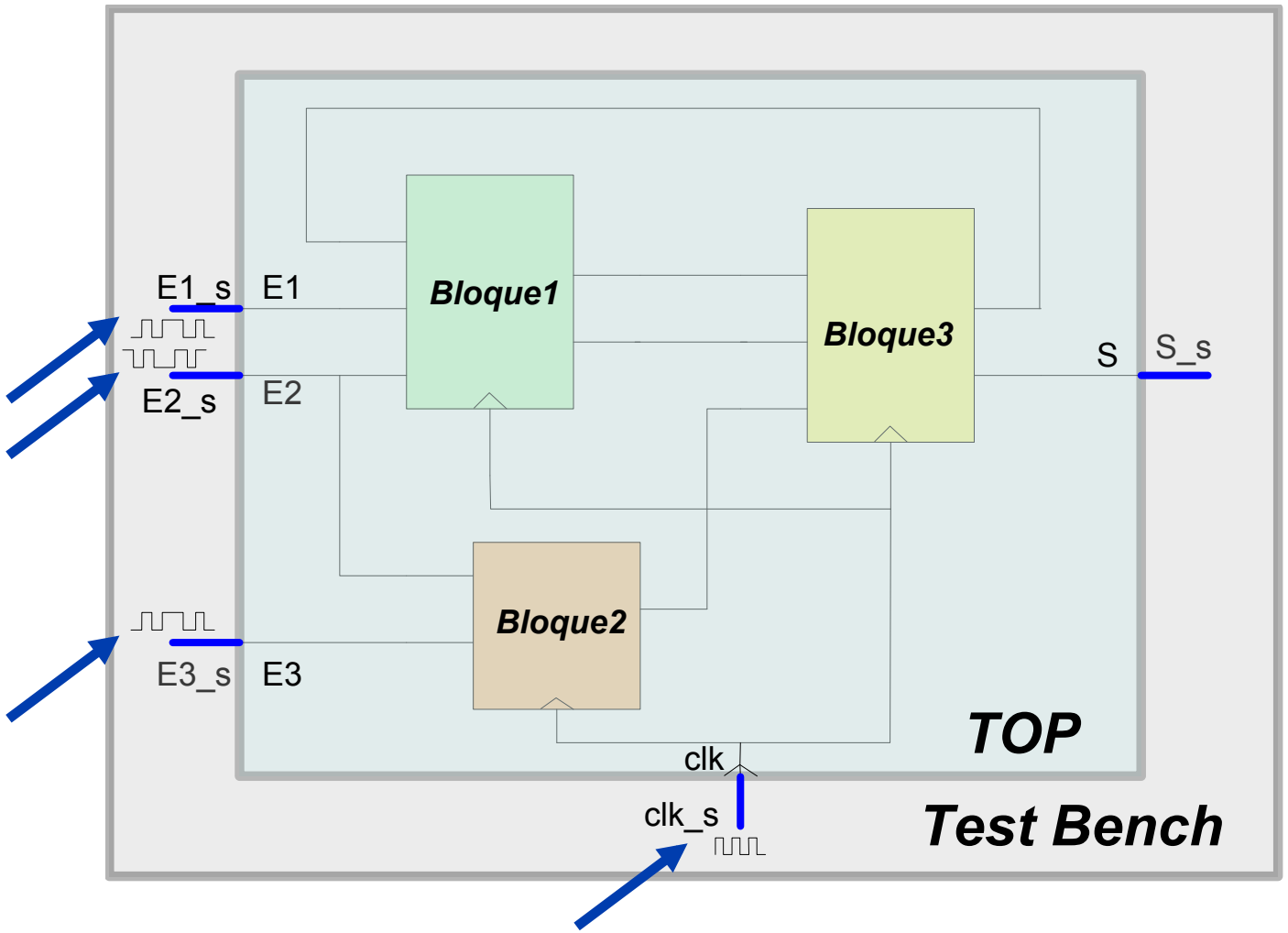
(el TestBench)

Diseño de un *test bench*



Se declara una señal por cada uno de los puertos

Diseño de un *test bench*



En la architecture del testbench se asignan estímulos a las entradas

Tema 4:

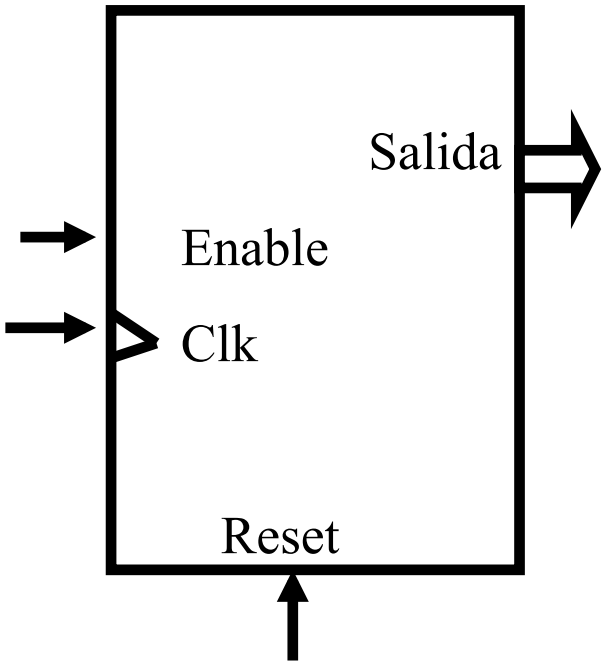
Simulación con un banco de pruebas VHDL - *test bench*.

4.1 Introducción

4.2 Diseño de un *test bench*

4.3 Ejemplos

Ejemplo

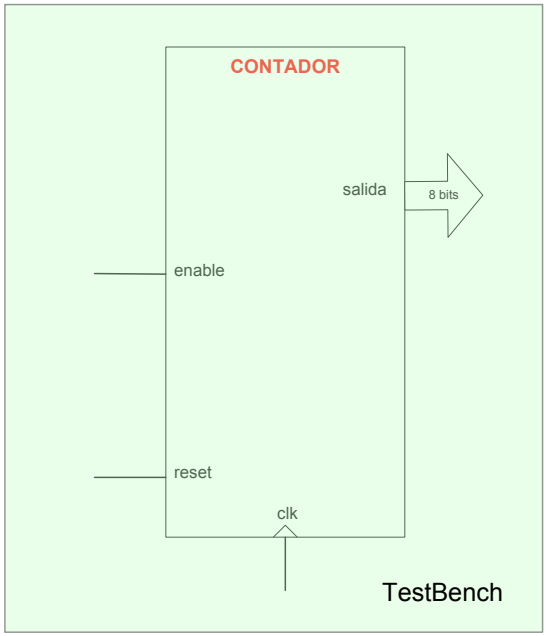



```
ENTITY contador IS
PORT (
salida: out std_logic_vector(7 DOWNTO 0);
clk: in std_logic;
reset: in std_logic;
enable: in std_logic
);
END contador;
```

Ejemplo

```
Architecture contador_arch of contador is
signal cuenta, p_cuenta: std_logic_vector(7 downto 0);
begin
   salida<=cuenta;
   comb:process(cuenta, enable)
   Begin
      if enable='1' then
         p_cuenta<=cuenta+1;
      Else
         p_cuenta<=cuenta;
      end if;
   end process;
   sinc:process(clk, reset)
   begin
      if reset='1' then
         cuenta <= (others => '0');
      esif(clk='1' and clk'event) then
         cuenta <= p_cuenta;
      end if;
   end process;
End contador_arch;
```

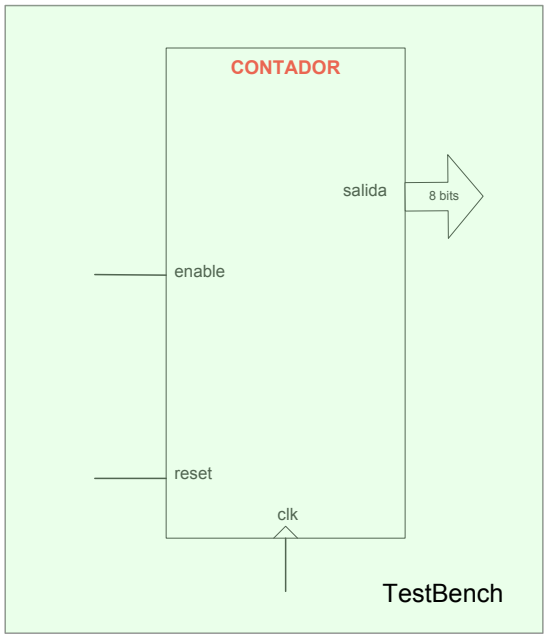
Ejemplo




```
ENTITY TestBench IS
END TestBench;
```

PASO 1: Definir la entidad TestBench sin añadir ningún puerto

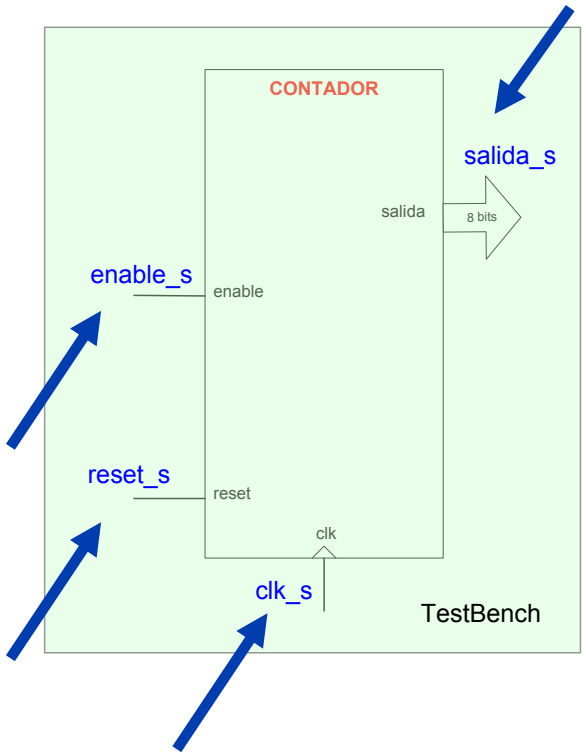
Ejemplo




```
ARCHITECTURE  TestBench_arch OF TestBench IS
COMPONENT contador
PORT (
  salida: out std_logic_vector(7 DOWNTO 0);
  clk: in std_logic;
  reset: in std_logic;
  enable: in std_logic
 );
END COMPONENT;
```

PASO 2: Declarar como componente el circuito para simular

Ejemplo



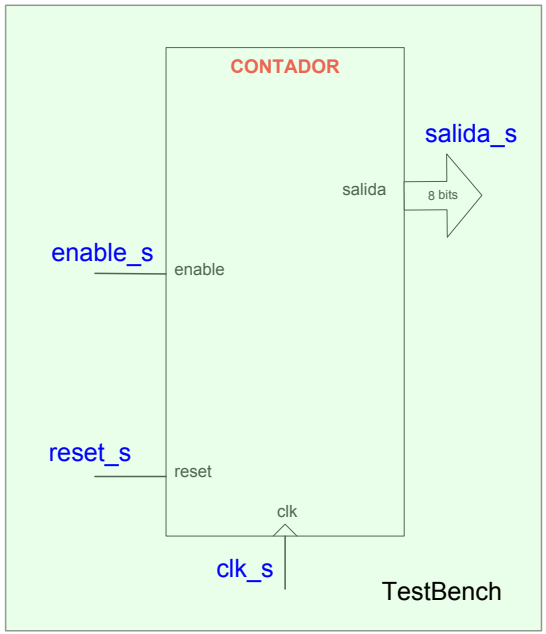
Valor por defecto (tiempo 0)

```
SIGNAL salida_s : std_logic_vector (7 downto 0);
SIGNAL clk_s : std_logic:= '0';
SIGNAL enable_s : std_logic;
SIGNAL reset_s: std_logic;

BEGIN   - - contenido de la arquitectura
```

PASO 3: Declarar una señal por cada puerto del componente.

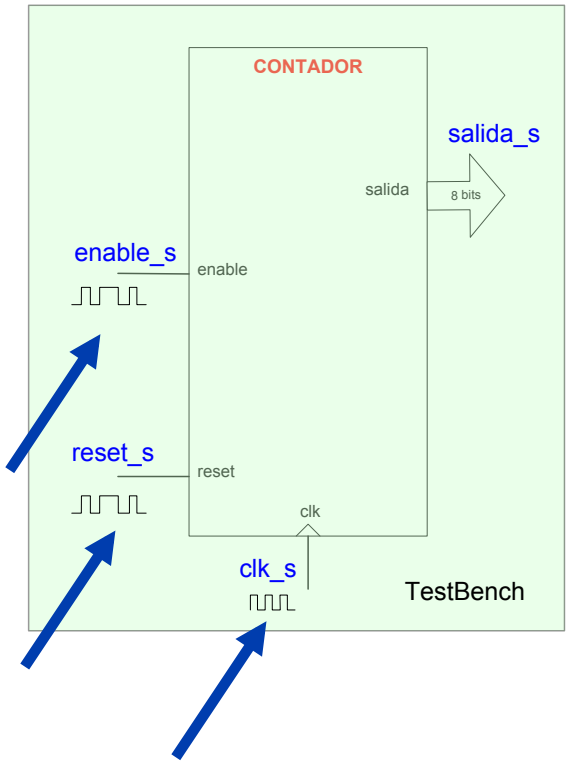
Ejemplo




```
DUT: contador
  PORT MAP(
    salida => salida_s,
    clk => clk_s,
    enable => enable_s,
    reset => reset_s);
```

PASO 4: Realizar una instancia al componente, conectando sus puertos a las señales correspondientes

Ejemplo




```
clk_s <= NOT clk_s AFTER 50 ns;

estimulos: PROCESS
BEGIN
  reset_s<='1';
  enable_s<='1';
  WAIT FOR 100 ns;
  reset_s<='0';
  WAIT FOR 1000 ns;
  enable_s<='0';
  WAIT;
END PROCESS;

END TestBench_arch;
```

PASO 5: Generar los estímulos

Formas de estímulos básicos

- Valor initial

SIGNAL nombre: TIPO := VALOR_INITIAL;

- Asignaciones en tiempo

Senal <= val1, val2 after XX, val3 after YY …;

- Asignaciones recurrentes en tiempo

Senal <= OP senal after XX ;

Estímulos en procesos

- Se puede utilizar la orden WAIT dentro de un *proceso sin lista sensible*
- Mientras el proceso esta parado en un "wait", las últimas asignaciones toman efecto.

WAIT for TIEMPO;

WAIT until CONDICION;

WAIT on SEÑAL;

Exemplo de procesos

```
process
Variable v: integer range 0 to 255;
Begin
  a <= '0';
  v := 1;
  wait for 100 ns;
  v := v + 1;
  a <= '1';
  wait on c, d;
  a <= '0';
  wait until d = 15 and d'event;
  a <= '1';
  wait on c for 200 ns;
  a <= v;
….
End process;
```

Acceso a ficheros

- Las librerías básicas contienen funciones simples para acceso a ficheros.
  - Es posible leer estímulos desde ficheros
  - Es posible salvar resultados en ficheros

```
LIBRARY STD;
USE STD.TEXTIO.ALL
LIBRARY IEEE;
USE IEEE.STD_LOGIC_TEXTIO.ALL
```

- ¡Cambio de sintaxis en 1993!

Acceso a ficheros

- Sintaxis original

  FILE fin : TEXT IS IN “i.txt”;

  FILE fout: TEXT IS OUT “o.txt”;

- Sintaxis VHDL’93

  FILE fin : TEXT open READ_MODE is “i.txt";

  FILE fout : TEXT open WRITE_MODE is “o.txt";

Lectura de ficheros

- Solamente se puede leer lineas enteras

  Variable línea: LINE;

  Readline (fin, linea);

- Se puede sacar los campos de la línea uno por uno:

  Read(linea, primer_campo);

  Read(linea, segundo_campo);

- Campos escritos en decimal o formato

  1 2

  3 4

  16#FF# 1

Escritura de ficheros

- Solamente se puede escribir líneas enteros

  Variable linea: LINE;

  writeline (fout, linea);

- Se construye la línea campo por campo

  write(linea, primer_campo);

  write(linea, “ “);

  write(linea, segundo_campo);

- Campos son escrito en su formato natural
- Normalmente existe las funciones HWRITE para escribir en HEX y BWRITE para escribir en binario.

Acceso a ficheros -- ejemplo

```
PROCESS (load)
VARIABLE lineain, lineaout: LINE;
VARIABLE campo1, campo2: integer;
 BEGIN
        IF (load = '1') THEN
           READLINE(fin, lineain);
           READ(lineain, campo1);
           READ(lineain, campo2);
           entrada1 <= campo1;
           entrada2 <= campo2;
        END IF;
        campo1 := conv_integer(senal);
        WRITE(lineaout, campo1);
        WRITELINE(fout, lineaout);
END PROCESS;
```