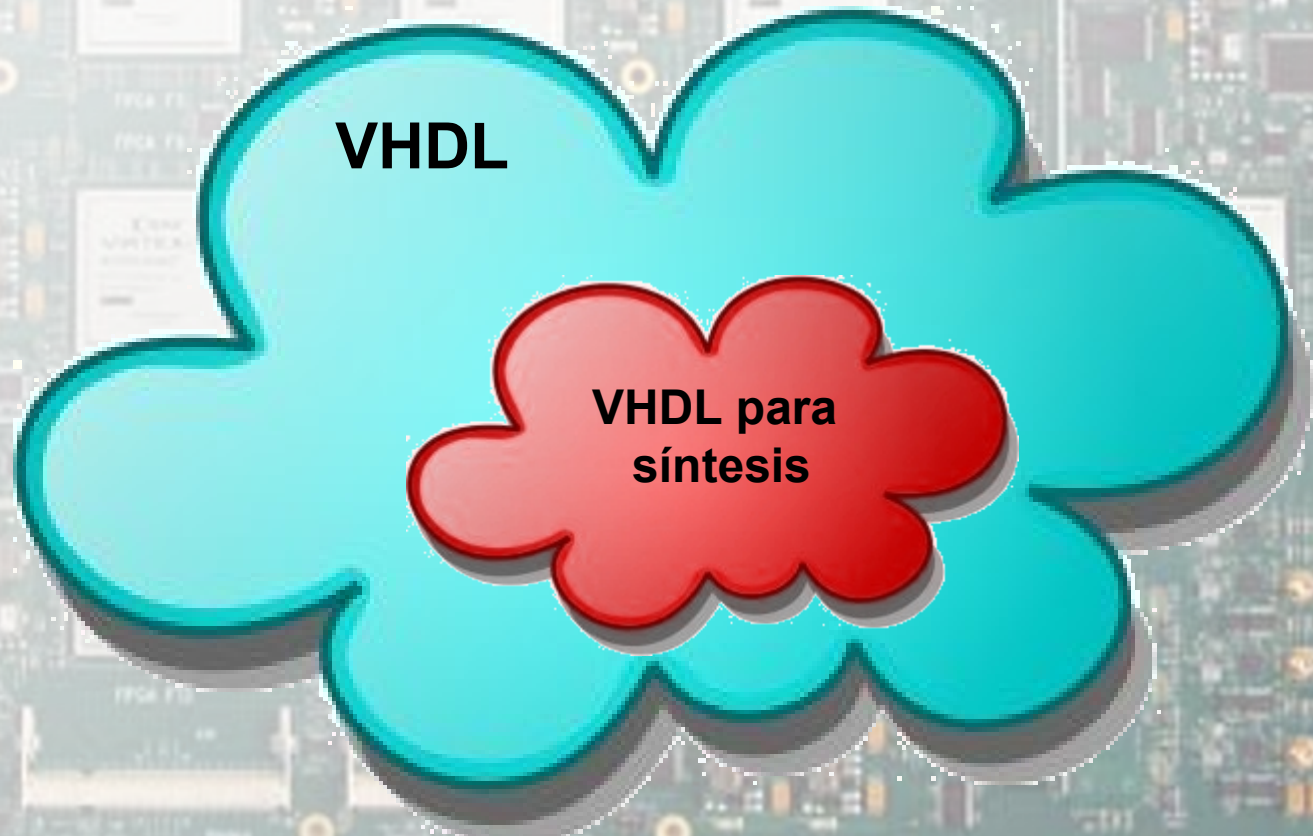


VHDL para síntesis

VHDL para síntesis

No siempre un código VHDL es sintetizable.
Sólo un subconjunto reducido del lenguaje se
utiliza para implementar circuitos.



VHDL para síntesis

Algunos consejos para realizar descripciones sintetizables

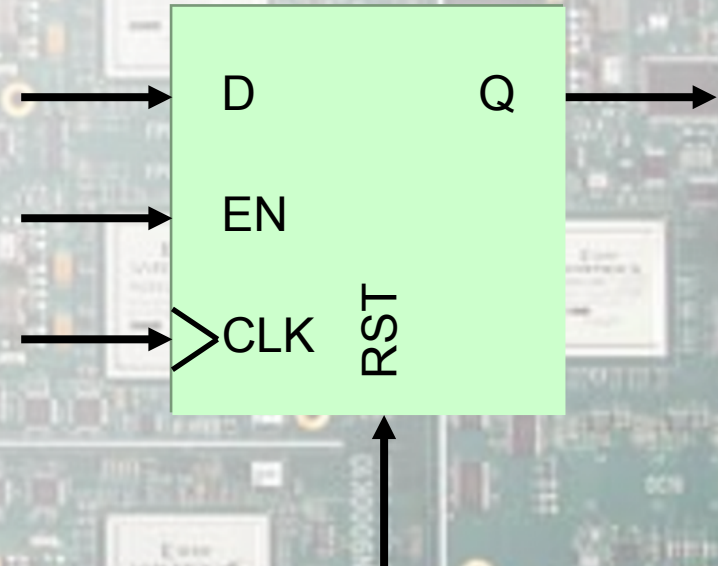
- No utilizar sentencias **wait**
- Evitar anidamiento excesivo en procesos combinacionales
- Utilizar **case** antes que varios **if**
- Utilizar un reloj por proceso
- No olvidar señales en las listas sensibles
- Evitar **if**'s en paralelo sobre mismas señales
- Al momento de optar entre descripciones diferentes, utilizar la más sencilla de interpretar



VHDL para síntesis

Flip flops tipo D con reset síncrono y clock enable

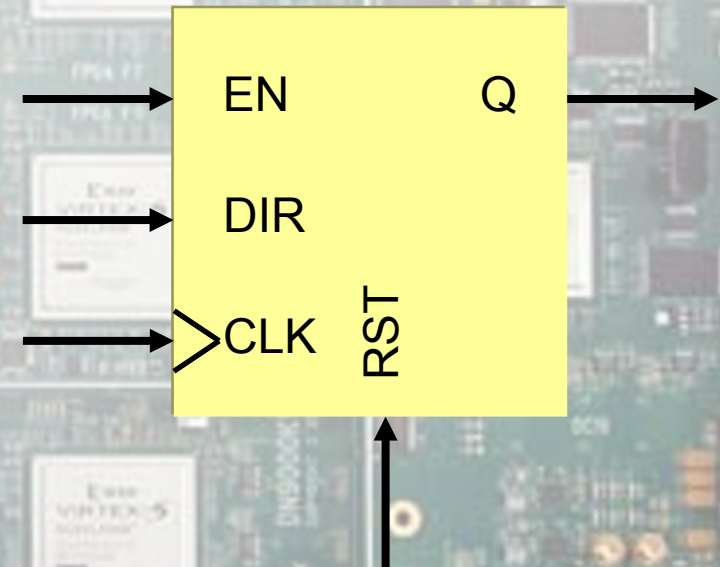
```
process(CLK)
begin
  if (CLK'event and CLK='1') then
    if RST='1' then
      Q <= '0';
    elsif (EN='1') then
      Q <= D;
    end if;
  end if;
end process;
```



VHDL para síntesis

Contadores up - down

```
process(CLK)
  variable Q_int : std_logic_vector(3 downto 0);
begin
  if (CLK'event and CLK='1') then
    if RST='1' then
      Q_int := (others => '0');
    elsif (EN='1') then
      if (DIR='1') then
        Q_int := Q_int + 1;
      else
        Q_int := Q_int - 1;
      end if;
    end if;
  end if;
  Q <= Q_int;
end process;
```

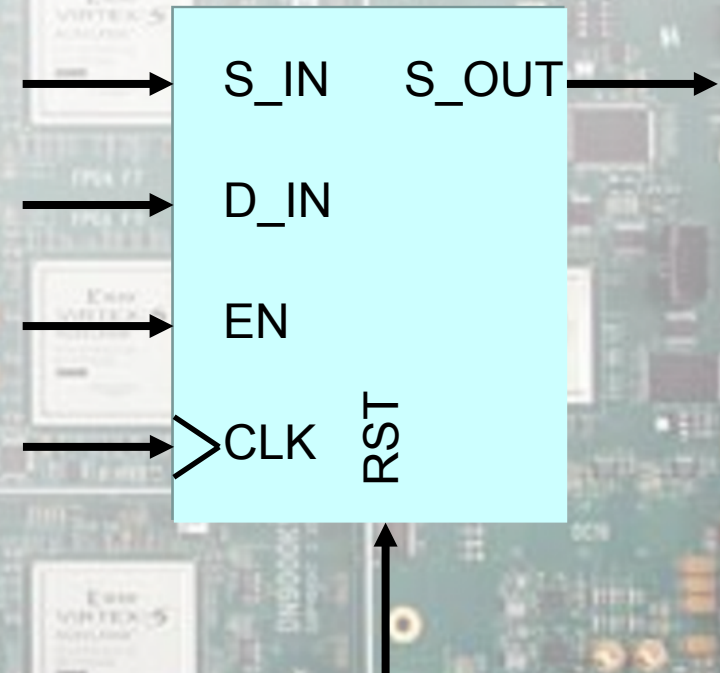


INTRODUCCIÓN A VHDL

VHDL para síntesis

Registros de desplazamiento con load y clock enable

```
process(CLK)
begin
  if (CLK'event and CLK='1') then
    if RST='1' then
      D <= (others => '0');
    elsif (EN='1') then
      if load='1' then
        D <= D_IN;
      else
        D <= D(6 downto 0)&S_IN;
      end if;
    end if;
  end if;
end process;
S_OUT <= D(7);
```

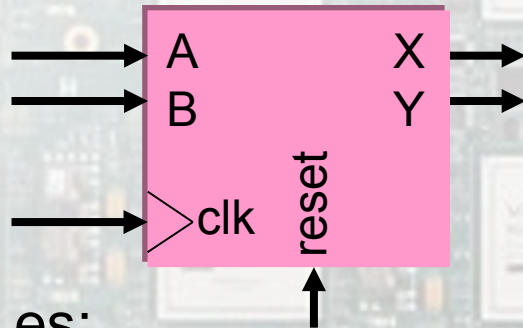


INTRODUCCIÓN A VHDL

VHDL para síntesis

Máquinas de estado de Moore

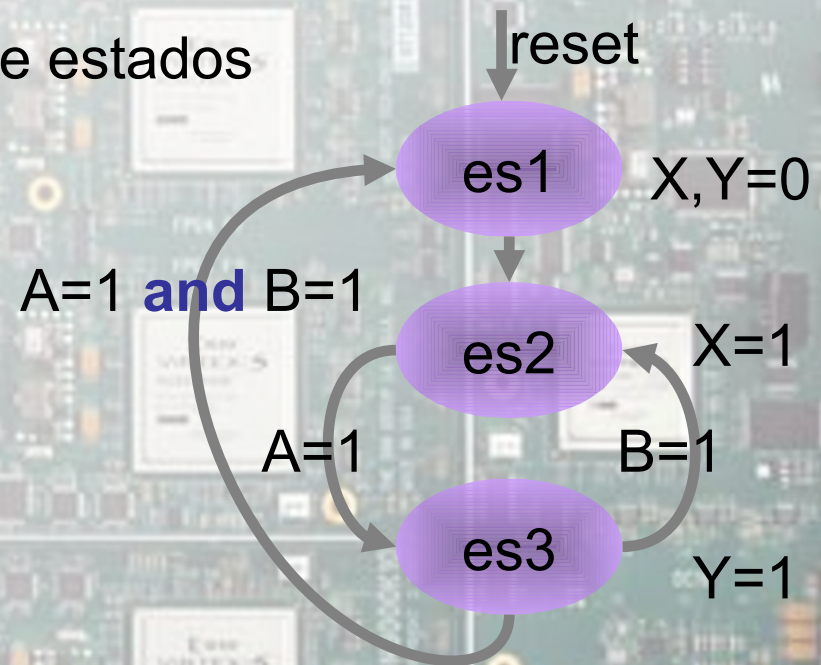
Supongamos la siguiente máquina de estados



La entidad es:

```

entity moore is
  port ( A,B      : in  std_logic;
        clk, reset : in  std_logic;
        X,Y       : out std_logic);
end entity moore;
  
```



INTRODUCCIÓN A VHDL

VHDL para síntesis

El esquema de una máquina de estados de Moore es el siguiente



VHDL para síntesis

Máquinas de estado de Moore

**Determinación
del estado
siguiente**

```
architecture behavioral of moore is  
  type estados is (es1, es2, es3);  
  signal actual, siguiente : estados;  
begin  
  ...
```

det_siguiente_estado:

```
process (A,B)
```

```
begin
```

```
  case actual is
```

```
    when es1 =>
```

```
      siguiente<= es2;
```

```
    when es2 =>
```

```
      if (A='1') then
```

```
        siguiente <= es3;
```

```
      else
```

```
        siguiente <= es2;
```

```
      end if;
```

```
    when es3 =>
```

```
      if (A='1' and B='1') then
```

```
        siguiente <= es1;
```

```
      elsif (B='1') then
```

```
        siguiente <= es2;
```

```
      else
```

```
        siguiente <= es3;
```

```
      end if;
```

```
    end case;
```

```
end process;
```

INTRODUCCIÓN A VHDL

VHDL para síntesis

Máquinas de estado de Moore

```
reg_estado_actual:  
process (clk)  
begin  
if (clk='1' and clk'event)  
then  
    if reset='1' then  
        actual<=es1;  
    else  
        actual<=siguiente;  
    end if;  
end process;
```

**Registro
de estado
actual**

INTRODUCCIÓN A VHDL

VHDL para síntesis

Máquinas de estado de Moore

```
process(actual)
begin
    case actual is
        when es1 =>
            X<='0';
            Y<='0';
        when es2 =>
            X<='0';
            Y<='0';
        when es3 =>
            X<='0';
            Y<='0';
    end case;
end process;
```

**Generación
de salidas**

VHDL para simulación

VHDL para simulación

EL lenguaje VHDL fue diseñado inicialmente como medio para el modelado y la simulación de sistemas digitales, por lo cual no impone limitaciones en cuanto a la simulación.

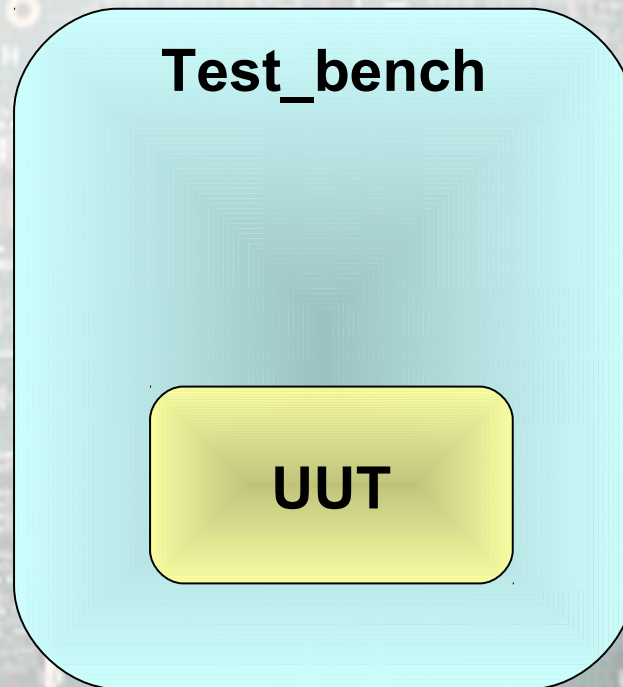
**VHDL
para simulación**



VHDL para simulación

El test bench (banco de pruebas)

Un test bench se encarga de generar los estímulos necesarios para comprobar un circuito y corroborar los resultados de la simulación.



Un testbench es una entidad sin puertos que se utiliza para probar un diseño

VHDL para simulación

Pasos para crear un testbench

1. Crear una entidad sin puertos
2. Instanciar la unidad bajo prueba en la arquitectura de dicha entidad
3. Declarar una señal por cada puerto de la señal bajo prueba con el mismo nombre. (e inicializar las entradas a algún valor)
4. Conectar las señales homónimas.
5. Crear los estímulos (generalmente en un proceso).
6. Escribir las sentencias que comprobarán los resultados.

VHDL para simulación

La sentencia **assert**

Esta sentencia se utiliza para evaluar los resultados de la unidad bajo prueba y comprobar si se corresponden con los esperados. Su sintaxis es:

assert condición **report** “mensaje” **severity** nivel



Mensaje a mostrar si **NO**
se cumple la condición

- NOTE
- WARNING
- ERROR
- FAILURE

INTRODUCCIÓN A VHDL

VHDL para simulación

Esquema básico para la generación de estímulos y comprobación de resultados

1

señal <= valor

2

wait for



3

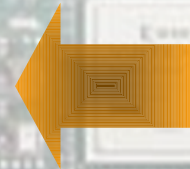
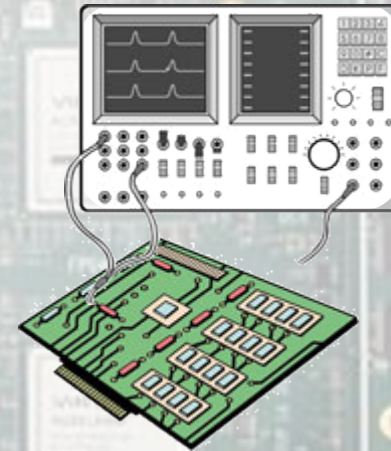
assert

resultado=esperado

VHDL para simulación

Ejemplo de testbench

```
entity testbench is  
end entity testbench  
architecture behavior of testbench is  
  component dff  
    port( D : in  std_logic;  
          Q : out std_logic;  
          clk : in  std_logic);  
  end component;  
  signal D,Q,clk std_logic;  
begin  
  uut : dff  
  port map( D =>D,  
            Q =>Q,  
            clk =>clk);
```



Instanciación de la
unidad bajo prueba

INTRODUCCIÓN A VHDL

VHDL para simulación

tb:

```
process()  
begin  
    D<='0';  
    wait until rising_edge(clk);  
    assert Q='0' report "Falla" severity error;  
    D<='1';  
    wait until rising_edge(clk);  
    assert Q='1' report "Falla" severity error;  
    wait;
```



```
end process;
```

clk_gen:

```
process()  
    clk <='0';  
    wait for 0.5 us;  
    clk <='1';  
    wait for 0.5 us;
```

```
end process;
```

```
end architecture behavior;
```

 Proceso de prueba Proceso para la
generación de reloj