DESCRIPCION EN VHDL DE CIRCUITOS LÓGICOS COMBINATORIOS

FUNDAMENTOS DE DISEÑO DIGITAL
DISEÑO DE SISTEMAS DIGITALES
OPTATIVA I. ISISA

Diseño (Programación) de una Estructura Básica Combinatoria

Declaración *Entidad*

Declaración *Arquitectura*

Sintaxis:

architecture *nombre_arquitectura* **of** *nombre_entidad* **is**
begin
    {***Enunciados Concurrentes***}
end [*nombre_arquitectura*]

Unidad de Cómputo/Cálculo que realiza lo siguiente:
- Lectura de Señales.
- Realiza cálculos basados en los valores de las Señales.
- Asigna los valores calculados a Señales específicas.

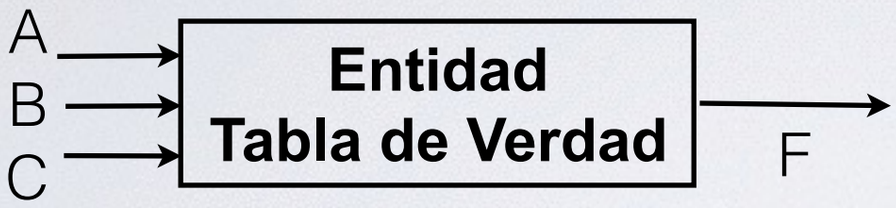
Enunciados Concurrentes

Asignación de Señal : Permite asignar un valor calculado a una señal o puerto.

Tipos:

- Asignaciones Condicionales de Señales – La construcción **when-else**
- Asignaciones de Señales mediante **Ecuaciones Booleanas**
- Asignaciones de Señales por Selección – La construcción **with-select-when**
- Ciclo que repite n veces asignaciones condicionales de señales, de señales mediante ecuaciones Booleanas o de señales por selección- **For Generate**

Asignaciones Condicionales de Señales - La construcción **when-else**



A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

```
LIBRARY IEEE;
USE IEEE.STAD_LOGIC_1164.ALL;

ENTITY TABLA IS

PORT  (A,B,C: IN STD_LOGIC;
           F:OUT STD_LOGIC
           );
END TABLA;

ARCHITECTURE A_TABLA OF TABLA IS
BEGIN

F<= ‘1’ WHEN ( A=’0’ AND B=’0’ AND C=’0’) ELSE
      ‘1’ WHEN ( A=’0’ AND B=’1’ AND C=’1’) ELSE
      ‘1’ WHEN ( A=’1’ AND B=’1’ AND C=’0’) ELSE
      ‘1’ WHEN ( A=’1’ AND B=’1’ AND C=’1’) ELSE
      ‘0’;

END A_TABLA;
```

La construcción **when-else** permite definir paso a paso el comportamiento de un sistema. Para esto, se declaran los valores que se deben asignar a una señal (o grupo) en función de las diferentes condiciones de entrada posibles. El orden en el que se declaren las condiciones de entrada, no es importante.

Asignaciones de Señales por Selección – La construcción **with-select-when**

A(1)	A(0)	F
0	0	1
0	1	0
1	0	1
1	1	1

```
LIBRARY IEEE;
USE IEEE.STAD_LOGIC_1164.ALL;

ENTITY TABLA IS

PORT  (A: IN STD_LOGIC_VECTOR (1 DOWNTO 0);
          F:OUT STD_LOGIC
          );
END TABLA;

ARCHITECTURE A_TABLA OF TABLA IS
BEGIN

WHIT A SELECT

F<= ‘1’ WHEN “00”,
      ‘0’ WHEN “01”,
      ‘1’ WHEN “10”
      ‘1’ WHEN OTHERS;

END A_TABLA;
```

- La estructura **with-select-when** se utiliza para asignar un valor (de varios posibles) a una señal o grupo de señales con base a los diferentes valores de otra señal o grupo de señales previamente seleccionada(o).

- Por lo general, un grupo de señales forman un vector, como en el ejemplo descrito *a(1)* y *a(0)* forman el vector *a.*

Únicamente, se utiliza la *coma* (**,**), el *punto y coma* (**;**) se utiliza cuando se finaliza la construcción ***with-select***

Diseño (Programación) de una Estructura Básica Combinatoria

Declaración *Entidad*

Declaración *Arquitectura*

Sintaxis:

architecture *nombre_arquitectura* **of** *nombre_entidad* **is**
begin

Process {lista sensitiva}
{*Enunciados secuenciales*}
end process;

end [*nombre_arquitectura*]

Proceso (**process**)
Permite definir un algoritmo secuencial que lee valores de Señales y calcula nuevos valores que son asignados a otras Señales.

Procesos (process)

- Cada proceso es conformado por un conjunto de enunciados secuenciales.

- Enunciados Secuenciales Son interpretados por la herramienta de síntesis en forma secuencial, es decir, uno por uno; por lo que el orden en el cual son declarados tiene un efecto significativo en la lógica que se intenta describir o sintetizar.

Enunciados Secuenciales →
- Enunciados ***if***
- Enunciados ***case***
- Enunciados ***loop***

Nota importante:
Una ***señal*** que se vea involucrada dentro de un proceso no recibe inmediatamente el valor asignado, sólo hasta el final del mismo. Una ***variable*** que sea utilizada dentro de un proceso sí recibe el valor de forma inmediata.

Enunciados if:
La construcción **if-then-else**

```
if la_condición_es_cierta then
    {ejecuta grupo-1 de enunciados secuenciales};
else
    {ejecuta grupo-2 de enunciados secuenciales};
end if;
```

Enunciados if:
La construcción **if-then-elsif-then-else**

```
if la_condición-1_se_cumple then
    {ejecuta grupo-1 de enunciados secuenciales};
elsif la_condición-2_se_cumple then
    {ejecuta grupo-2 de enunciados secuenciales};
else
    {ejecuta grupo-3 de enunciados secuenciales};
end if;
```

Operadores Relacionales

CARACTERISTICAS

- Uso: para fines de comparación de datos
- Operadores incluidos en los paquetes: std_numeric y std_logic_arith
- Los operadores de igualdad y desigualdad (=,/=) utilizan todos los tipos de datos
- Los operadores (>, <, >= y <=) son definidos para los tipos escalar y arreglos unidimensionales de tipos de datos enumerados o enteros

Operador	Significado
=	igual
/=	diferente
<	menor
<=	menor igual
>	mayor
>=	mayor igual

Enunciados CASE:

La construcción **case - when** ejecuta una o varias instrucciones secuenciales que dependen del valor de una sola expresión.

SINTAXIS

```
case expresion is
        when caso => enunciados secuenciales;
        when others => enunciados secuenciales;
end case;
```

EJEMPLO:

```
case puntuacion of
      when 9 to 10        => acta <=“Sobresaliente”;
      when 8 downto 7  => acta <=“Notable”;
      when 5  |  6    => acta <=“Aprobado”;
      when 0          => acta <=“No presentado”;
      when others   => acta <=“Suspenso”;
end case;
```

FOR-LOOP-GENERATE: La sentencia o declaración LOOP o GENERETE es usada siempre que una operación necesita ser repetida.

FOR LOOP → DENTRO DE PROCESOS

Sintaxis

```
FOR VAR IN RANGO LOOP
------------------
------------------   CODIGO
------------------
END LOOP;
```

Ejemplos:

```
FOR I IN 0 TO 3 LOOP
------------------  CODIGO
END LOOP;

FOR I IN 3 DOWNTO 0 LOOP
------------------  CODIGO
END LOOP;
```

FOR LOOP → DENTRO DE PROCESOS

Sintaxis

```
FOR VAR IN RANGO LOOP
------------------
------------------   CODIGO
------------------
END LOOP;
```

Ejemplos:

```
FOR I IN 0 TO 3 LOOP
------------------  CODIGO
END LOOP;

FOR I IN 3 DOWNTO 0 LOOP
------------------  CODIGO
END LOOP;
```

OPERADORES DE CORRIMIENTO

➡ **SLL.** Desplazamiento lógico hacia la izquierda, llenado con ceros.

➡ **SRL**. Desplazamiento lógico hacia la derecha, llenado con ceros.

➡ **SLA**. Desplazamiento aritmético hacia la izquierda, llenado con el bit de menor peso.

➡ **SRA**. Desplazamiento aritmético hacia la derecha, llenado con el bit de mayor peso.

➡ **ROL**. Rotación a la izquierda.

➡ **ROR**. Rotación a la derecha.