Sesión 6: Diseño Lógico con VHDL

VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Estilos de descripción de un circuito lógico

- **Comportamiento**
- **Flujo de Datos (ó Transferencia de Registros)**
- **Estructural**

Ejemplo.

Un sistema secuencial síncrono, que llamaremos *dos_con* (dos consecutivos) que produce salida '1' cuando aparecen dos o más valores iguales consecutivos en su entrada.

Diagramas de bloques y de estados:

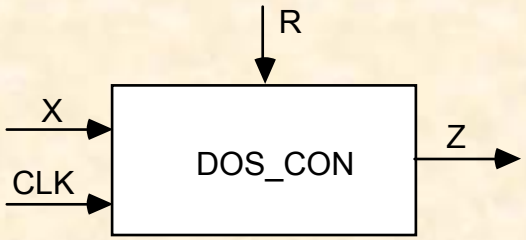


Diagrama de Bloques

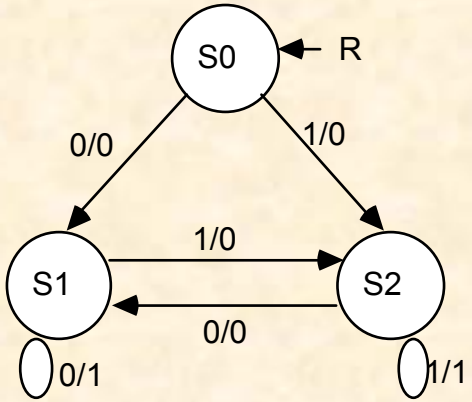


Diagrama de Estados

Diagrama de Bloques --> Declaración de entidad:

```
ENTITY dos_con IS
  PORT(clk, r, x : IN bit; z : OUT bit);
END dos_con;
```

Diagrama de estados --> lo traduciremos a diferentes arquitecturas.

VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Comportamiento

- Define la funcionalidad del dispositivo con un algoritmo secuencial, sin referencia alguna a su implementación.

- En nuestro caso escribiremos un algoritmo que sigue el comportamiento de su tabla de estados:

	X	
Estado	0	1
S0	S1/0	S2/0
S1	S1/1	S2/0
S2	S1/0	S2/1

```
ARCHITECTURE algorítmica OF dos_con IS
  TYPE estado IS (s0, s1, s2);
  SIGNAL q : estado := s0;
BEGIN
  PROCESS(r, x, clk, q)
  BEGIN
    IF (r'EVENT AND r = '0') THEN q <= s0;
    ELSIF (clk'EVENT AND clk = '1') THEN
     IF x = '0' THEN q <= s1; ELSE q <= s2; END IF;
    END IF;
    IF (q'EVENT OR x'EVENT) THEN
      IF (q = s1 AND x = '0') OR (q = s2 AND x = '1') THEN
        z <= '1';
      ELSE
        z <= '0';
      END IF;
    END IF;
  END PROCESS;
END algorítmica;
```


VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Flujo de datos

- Representa a un sistema digital con un conjunto concurrente de ecuaciones que expresan el flujo de información por los módulos del sistema.

Codificación de estados

	Código
Estado	y1 y0
S0	0 0
S1	0 1
S2	1 1

```
ARCHITECTURE flujo OF dos_con IS
  SIGNAL y1, y0 : bit;
BEGIN
  estado : BLOCK((clk = '1' AND NOT clk'STABLE) OR r = '0')
  BEGIN
    y1 <= GUARDED '0' WHEN r = '0' ELSE x;
    y0 <= GUARDED '0' WHEN r = '0' ELSE '1';
  END BLOCK estado;
  z <= y0 AND ((NOT y1 AND NOT x) OR (y1 AND x));
END flujo;
```

Tabla binaria:

	X	
y1 y0	0	1
0 0	0 1 / 0	1 1 / 0
0 1	0 1 / 1	1 1 / 0
1 1	0 1 / 0	1 1 / 1
1 0	- - / -	- - / -
	Y1 Y0 / Z	

VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Estructural

- Una descripción estructural declara los componentes del dispositivo y expresa sus interconexiones.

- Esquemático de dos_con compuesto de biestables y puertas:

Y1

0	1	1	0
-	-	1	0

Y1 = x

Y0

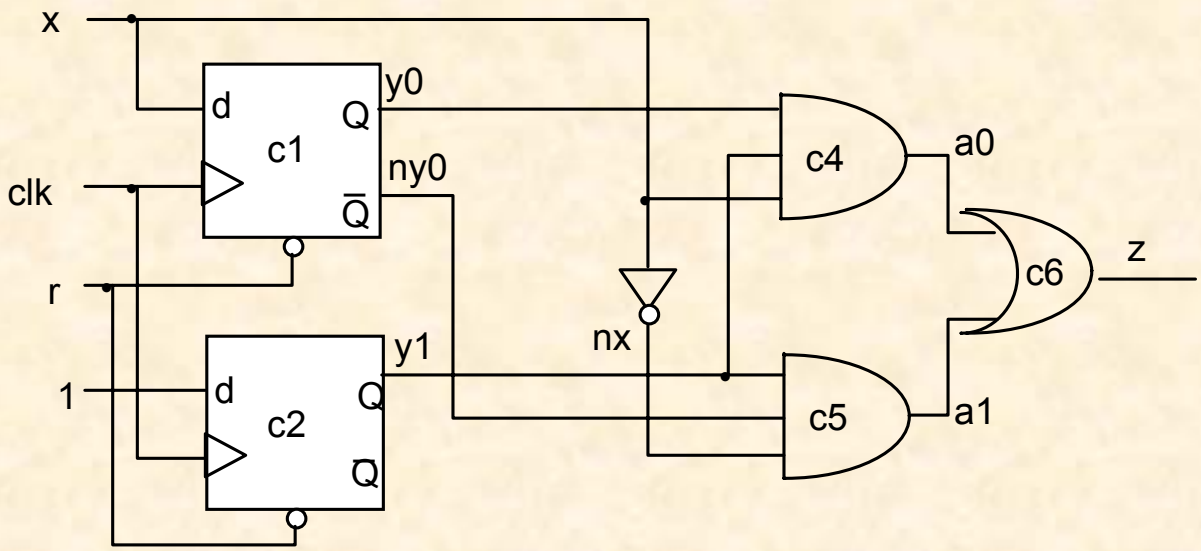
1	1	1	1
-	-	1	1

Y0 = 1

Z

0	0	0	1
-	-	1	0

$$Z = \overline{y_1} y_0 \overline{x} + y_1 \overline{y_0} x$$

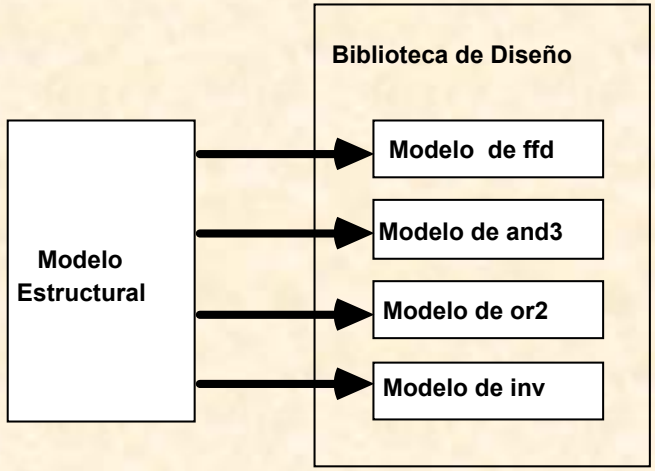


Modelo de flip-flop d (ffd)

```
ENTITY ffd IS
  PORT(clk, d, nclr : IN bit; q, qn : OUT bit);
END ffd;
ARCHITECTURE comportamiento OF ffd IS
BEGIN
  bffd : BLOCK((clk = '1' AND NOT clk'STABLE) OR nclr = '0')
  BEGIN
    q <= GUARDED '0' WHEN nclr = '0' ELSE d;
    qn <= GUARDED '1' WHEN nclr = '0' ELSE NOT d;
  END BLOCK bffd;
END comportamiento;
```

Modelo de inversor (inv)

```
ENTITY inv IS
  PORT(e : IN bit; s : OUT bit);
END inv;
ARCHITECTURE comportamiento OF inv IS
BEGIN
  s <= NOT e;
END comportamiento;
```



Modelo de puerta and de tres entradas (and3)

```
ENTITY and3 IS
  PORT(e1, e2, e3 : IN bit; s : OUT bit);
END and3;
ARCHITECTURE comportamiento OF and3 IS
BEGIN
  s <= e1 AND e2 AND e3;
END comportamiento;
ortamiento;
```

Modelo de puerta or de dos entrdas (or2)

```
ENTITY or2 IS
  PORT(e1, e2 : IN bit; s : OUT bit);
END or2;
ARCHITECTURE comportamiento OF or2 IS
BEGIN
  s <= e1 OR e2;
END comportamiento;
```

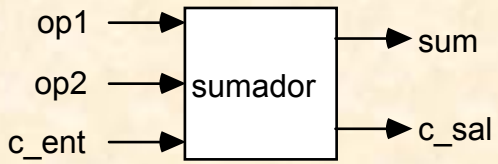
Arquitectura Estructural

```
USE WORK.ALL;
ARCHITECTURE estructural OF dos_con IS

-- declaración de señales de interconexión
  SIGNAL y0, y1, a0, a1 : bit := '0';
  SIGNAL ny0, nx : bit := '1';
  SIGNAL uno : bit := '1';

-- declaración de componentes
  COMPONENT ff_d
    PORT(clk, d, nclr : IN bit; q, qn : OUT bit);
  END COMPONENT;
  COMPONENT invg
    PORT(e : IN bit; s : OUT bit);
  END COMPONENT;
  COMPONENT and3g
    PORT(e1, e2, e3 : IN bit; s : OUT bit);
  END COMPONENT;
  COMPONENT or2g
    PORT(e1, e2 : IN bit; s : OUT bit);
  END COMPONENT;

-- especificación de configuración
  FOR ALL : ff_d USE ENTITY ffd(comportamiento);
  FOR ALL : invg USE ENTITY inv(comportamiento);
  FOR ALL : and3g USE ENTITY and3(comportamiento);
  FOR ALL : or2g USE ENTITY or2(comportamiento);
```

```
-- instanciación de componentes

BEGIN
  c1 : ff_d   PORT MAP(clk, x, r, y0, ny0);
  c2 : ff_d   PORT MAP(clk, uno, r, y1, OPEN);
  c3 : invg   PORT MAP(x, nx);
  c4 : and3g  PORT MAP(x, y0, y1, a0);
  c5 : and3g  PORT MAP(ny0, y1, nx, a1);
  c6 : or2g   PORT MAP(a0, a1, z);
END estructural;
```


VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Un segundo ejemplo



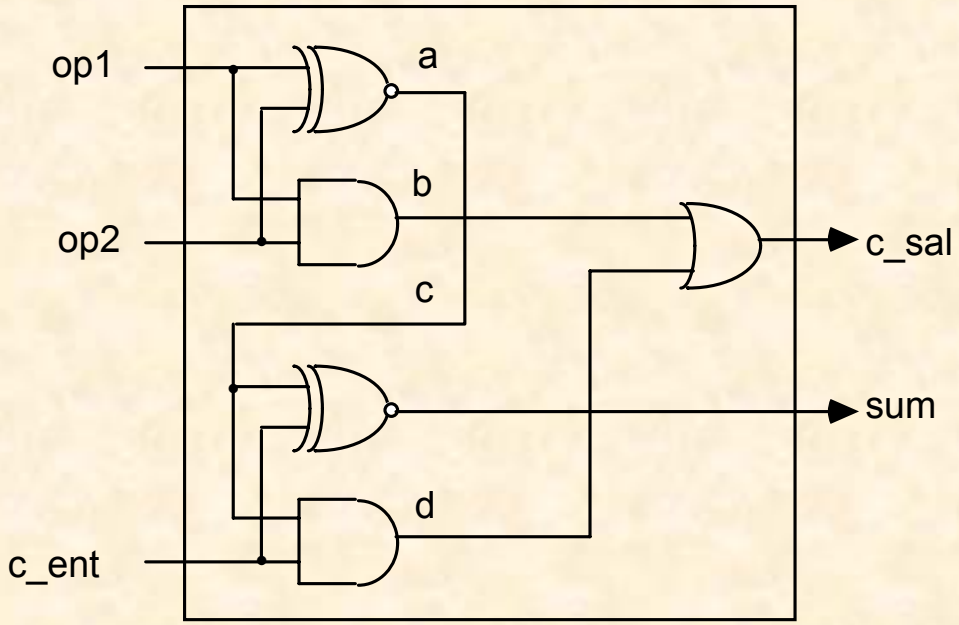

```
ENTITY sumador IS
  PORT(op1, op2 : IN bit;
       c_ent : IN bit;
       sum : OUT bit;
       c_sal : OUT bit);
END sumador;
```

Arquitectura de *comportamiento* :

```
ARCHITECTURE comportamiento OF sumador IS
  BEGIN

 PROCESS(op1, op2, c_ent)
FUNCTION bit_int (b : bit) RETURN INTEGER IS
  VARIABLE e : INTEGER;
  BEGIN
    IF b = '1' THEN e := 1; ELSE e := 0; END IF;
    RETURN e;
  END bit_int;
FUNCTION bool_bit (e : BOOLEAN) RETURN bit IS
  VARIABLE b : bit;
  BEGIN
    IF (e = TRUE) THEN b := '1'; ELSE b := '0'; END IF;
    RETURN b;
END bool_bit;
    VARIABLE res : INTEGER;
    BEGIN
      res := bit_int(op1) + bit_int(op2) + bit_int(c_ent);
      sum <= bool_bit((res MOD 2) = 1) AFTER 12 NS;
      c_sal <= bool_bit(res > 1) AFTER 10 NS;
    END PROCESS;
END comportamiento;
```


VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

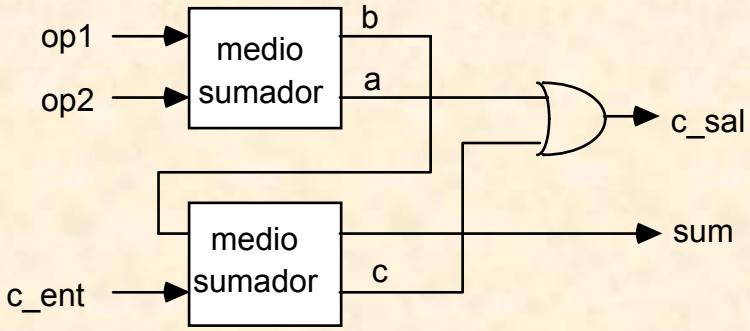
Flujo de Datos




```
ARCHITECTURE flujo_de_datos OF sumador IS
  SIGNAL a,b,c,d : BIT;
BEGIN
  a <= op1 XOR op2 AFTER 5 NS;
  b <= op1 AND op2 AFTER 2 NS;
  d <= a AND c_ent AFTER 2 NS;
  c_sal <= b OR d AFTER 2 NS;
  sum <= a XOR c_ent AFTER 5 NS;
END flujo_de_datos;
```


VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

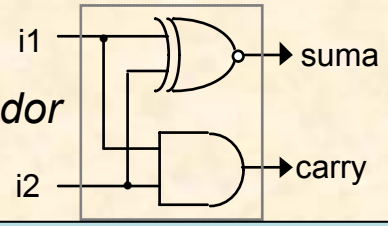
Estructural




```
USE work.ALL;
ARCHITECTURE estructural OF sumador IS
  COMPONENT medio_sumador
    PORT(i1, i2 : IN bit; suma, carry : OUT bit);
  END COMPONENT;
  COMPONENT or2
    PORT(i1, i2 : IN bit; o : OUT bit);
  END COMPONENT;
  SIGNAL a, b, c : bit;
BEGIN
  comp2 : medio_sumador PORT MAP(op1, op2, b, a);
  comp3 : or2  PORT MAP(a, c, c_sal);
  comp4 : medio_sumador PORT MAP(b, c_ent, sum, c);
END estructural;
```

COMPONENTES

Medio sumador



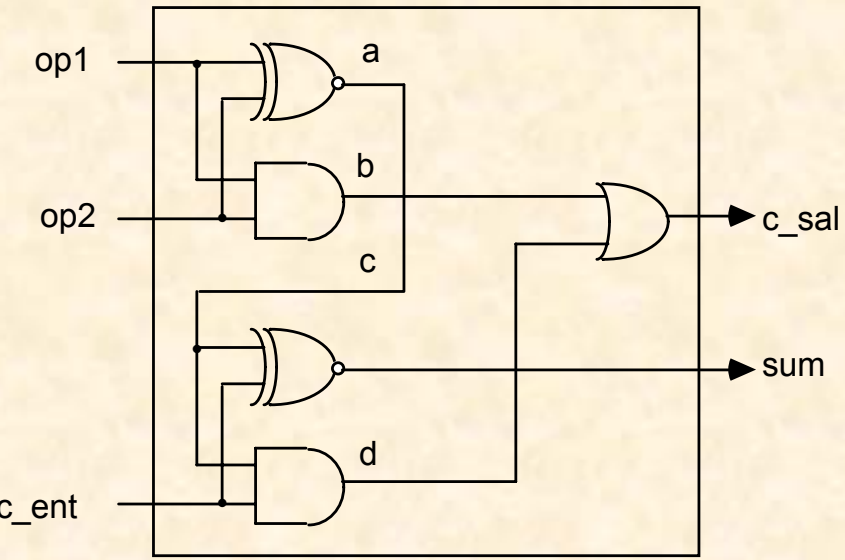

```
(modelo de flujo_de_datos:)
ENTITY medio_sumador IS
  PORT(i1, i2 : IN bit; suma, carry : OUT bit);
END medio_sumador;
ARCHITECTURE flujo_de_datos OF medio_sumador IS
BEGIN
  carry <= i1 AND i2;
  suma <= i1 XOR i2;
END flujo_de_datos;
```

Puerta or :

```
ENTITY or2 IS
  PORT(i1, i2 : IN bit; o : OUT bit);
END or2;
ARCHITECTURE comportamiento OF or2 IS
BEGIN
  o <= i1 OR i2;
END comportamiento;
```


VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Arquitectura Estructural en Términos de Puertas




```
USE work.ALL;
ARCHITECTURE puertas OF sumador IS
  COMPONENT or2
    PORT(i1, i2 : IN bit; o : OUT bit);
  END COMPONENT;
  COMPONENT and2
    PORT(i1, i2 : IN bit; o : OUT bit);
  END COMPONENT;
  COMPONENT xor2
    PORT(i1, i2 : IN bit; o : OUT bit);
  END COMPONENT;
  SIGNAL a, b, c, d : bit;
BEGIN
  comp1 : xor2  PORT MAP(op1, op2, a);
  comp2 : and2  PORT MAP(op1, op2, b);
  comp3 : xor2  PORT MAP(a, c_ent, sum);
  comp4 : and2  PORT MAP(a, c_ent, d);
  comp5 : or2  PORT MAP(b, d, c_sal);
END puertas;
```


VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Arquitectura Estructural Mixta

En VHDL podemos mezclar diferentes estilos de descripción, por ejemplo, el estructural y flujo de datos :

```
USE work.ALL;
ARCHITECTURE mezcla OF sumador IS
  COMPONENT medio_sumador
    PORT(i1, i2 : IN bit; suma, carry : OUT bit);
  END COMPONENT;
  SIGNAL a, b, c : bit;
BEGIN
  comp2 : medio_sumador PORT MAP(op1, op2, b, a);
  c_sal <= a OR c;
  comp4 : medio_sumador PORT MAP(b, c_ent, sum, c);
END mezcla;
```

Sistemas de valores lógicos en VHDL

En VHDL se pueden definir muchos sistemas de valores lógicos.
La elección depende de 3 factores:

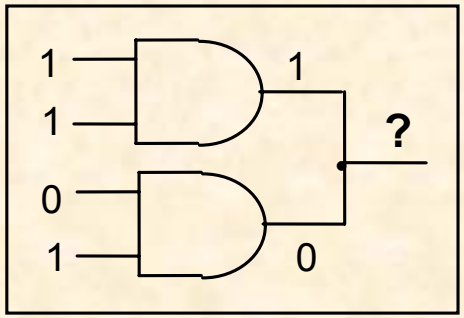
- *Nivel de resolución del modelo*.
  - En los modelos de comportamiento se utilizan los valores binarios.
  - En los modelos estructurales que incluyen factores dependientes de la tecnología de implementación, se utilizan sistemas multivaluados, con los valores suficientes para propagar todos los estados que resulten de interés en el proceso de diseño.
- *Portabilidad del modelo*.
  - El único sistema de valores lógicos definido en VHDL es el binario
  - Existen sistemas multivaluados que funcionan como sistemas estándar. Ejemplo: paquete STD_LOGIC_1164 que define un sistema de 9 valores.
- *Carga computacional*
  - Cuanto más complejo sea el sistema utilizado mayor será la carga computacional que recaerá sobre el simulador VHDL.
  - El sistema multivaluado no debería sobrepasar la complejidad exigida por el nivel de resolución del modelo.

VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Sistema de valores con 2 estados

```
TYPE bit IS ('0', '1');
TYPE bit_vector IS ARRAY (NATURAL RANGE <>) OF bit;
```

- La primera define el tipo binario básico (tipo *bit*)
- La segunda el tipo binario vectorial (tipo *bit_vector*).
- Este sistema resulta inoperante cuando se quiere modelar la posibilidad de que múltiples fuentes suministren valores conflictivos de la misma intensidad a una única señal.
- Por ejemplo, la conexión de las salidas de las dos puertas *AND* puede que sea correcta en determinadas tecnologías (salida en colector abierto)
- En otras se tratará sin duda de un error de diseño, y el sistema de dos estados no puede representar la condición de error en la señal de salida.



- El sistema de dos estados tampoco puede modelar el problema de los valores iniciales en puertas con retardo: hasta que transcurra el tiempo de retardo, los nodos de salida están en un estado desconocido (ni '1' ni '0').

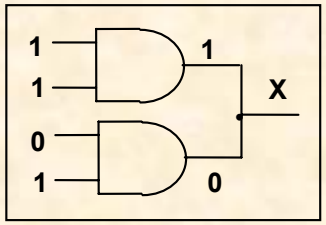
Sistema de valores con 3 estados

- Para superar las anteriores limitaciones del sistema binario se pueden introducir otros valores.
- Un valor *desconocido o indeterminado*, representado por 'X', se puede utilizar para resolver el problema de la conexión de más de una señal de salida con valores lógicos diferentes.
- El básico (*bitx01*) se declararía como tipo enumerado y el vectorial (*bitx01_vector*) como *array*

```
TYPE bitx01 IS  ('X','0', '1' );
TYPE bitx01_vector IS ARRAY (NATURAL RANGE <>) OF bitx01;
```

- Supondremos que el '0' y el '1' tienen la misma intensidad, por lo que en caso de conectar dos salidas que generen valores opuestos se transmitirá el valor indeterminado 'X'.
- Para ello será necesario la siguiente función de resolución:

```
FUNCTION busx01(entrada : bitx01_vector) RETURN bitx01 IS
  VARIABLE uno, cero : BOOLEAN := FALSE;
  BEGIN
    FOR i IN entrada'RANGE LOOP
                  CASE entrada(i) IS
                     WHEN '0' => IF uno THEN RETURN 'X'; ELSE cero := TRUE;END IF;
                     WHEN '1' => IF cero THEN RETURN 'X';ELSE uno := TRUE;END IF;
                     WHEN 'X' => RETURN 'X';
                  END CASE;
   END LOOP;
END busx01;
```



• El efecto de la función de resolución anterior se puede incorporar a la definición de dos tipos resueltos, uno para el básico (*bitx01res*) y otro para el vectorial (*bitx01res_vector*):

```
SUBTYPE bitx01res IS busx01 bitx01;
TYPE bitx01res_vector IS ARRAY (NATURAL RANGE <>) OF bitx01res;
```

• También habrá que incorporar al nuevo sistema trivaluado la sobrecarga del significado de los operadores lógicos (*AND*, *OR*, etc.) para las nuevas combinaciones de valores. Por ejemplo, para el operador *AND* podemos definir la siguiente función:

```
FUNCTION "AND" (a, b : bitx01) RETURN bitx01 IS
        TYPE tabla_bitx01 IS ARRAY (bitx01, bitx01) OF bitx01;
        CONSTANT tabla : tabla_bitx01 := (('X','0','X'), ('0','0','0'), ('X','0','1'));
BEGIN
          RETURN tabla (a,b);
END;
```

• Esta función implementa la tabla

a	b	AND (a,b)
X	X	X
X	0	0
X	1	X
0	X	0
0	0	0
0	1	0
1	X	X
1	0	0
1	1	1

VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Paquete completo

```
PACKAGE valx01 IS
TYPE bitx01 IS  ('X','0', '1' );
TYPE bitx01_vector IS ARRAY (NATURAL RANGE <>) OF bitx01;
FUNCTION busx01(entrada : bitx01_vector) RETURN bitx01;
SUBTYPE bitx01res IS busx01 bitx01;
TYPE bitx01res_vector IS ARRAY (NATURAL RANGE <>) OF bitx01res;
FUNCTION "AND" (a, b : bitx01) RETURN bitx01;
TYPE tabla_bitx01 IS ARRAY (bitx01, bitx01) OF bitx01;
CONSTANT tabla : tabla_bitx01 := (('X','0','X'), ('0','0','0'), ('X','0','1'));
END valx01;
```

```
PACKAGE BODY valx01 IS
FUNCTION busx01(entrada : bitx01_vector) RETURN bitx01 IS
                VARIABLE uno, cero : BOOLEAN := FALSE;
BEGIN
                FOR i IN entrada'RANGE LOOP
                    CASE entrada(i) IS
                                WHEN '0' => IF uno THEN RETURN 'X'; ELSE cero := TRUE;END IF;
                                WHEN '1' => IF cero THEN RETURN 'X'; ELSE uno := TRUE;END IF;
                                WHEN 'X' => RETURN 'X';
                                END CASE;
                END LOOP;
END busx01;

FUNCTION "AND" (a, b : bitx01) RETURN bitx01 IS
BEGIN
                RETURN tabla (a,b);
END;
END valx01;
```


VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997

Practica 6 : Diseño Lógico con VHDL

Objetivos:

Diseño de pequeños sistemas digitales utilizando los tres estilos de descripción VHDL: comportamiento, transferencia de registros o flujo de datos, y estructural.

Práctica a realizar:

Diseña un sistema secuencial síncrono con al menos cinco estados utilizando los tres estilos básicos de descripción en VHDL:

1. Comportamiento
2. Transferencia de registros o flujo de datos
3. Estructural

Notas:
1. Se deben modelar los retardos.
2. Se puede utilizar una lógica binaria o bien multivaluada

Resultados a entregar:

Documentación del diseño incluyendo:

1. Especificación completa y precisa del sistema modelado.
2. Listado VHDL comentado de los programas correspondientes a cada uno de los estilos de descripción.
3. Tests de entrada/salida que muestren el correcto funcionamiento del sistema en cada uno de los niveles de descripción. Incluir diagramas de tiempo.

VHDL: de la tecnología a la arquitectura de computadores. José J. Ruz Ortiz, Síntesis 1997