Curso a Distancia

INTRODUCCIÓN AL DISEÑO DIGITAL CON VHDL Y LÓGICA PROGRAMABLE

Ing. Arturo J. Miguel de Priego Paz Soldán

www.tourdigital.net

Chincha, Perú
30 de octubre de 2008

Capítulo 4

DESCODIFICADORES

Descodificadores binarios.
Modelos de circuitos integrados 7442, 74138 y 74139.
Descodificador 1 de 32 líneas.
Descodificador 1 de N líneas.
Descodificadores de siete segmentos.

Un descodificador activa una o varias salidas de acuerdo a un código de entrada. Los descodificadores empleados normalmente en el diseño lógico son descodificadores binarios y de siete segmentos. Típicamente, se utilizan para seleccionar unidades de un sistema (como chips de memoria o periféricos), para repartir datos de una fuente a varios destinos, y como generador de funciones lógicas.

Objetivos

Al finalizar este módulo estarás en capacidad de:

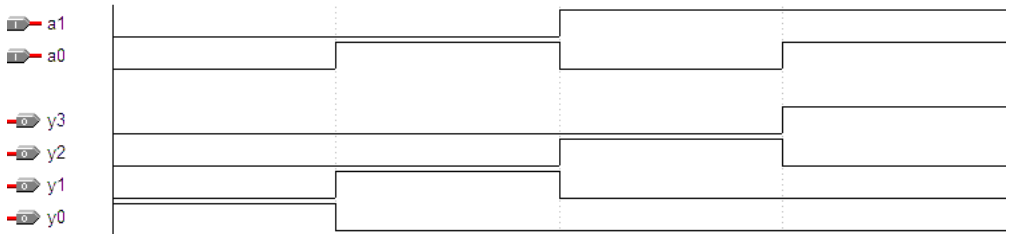
- ✓ Diseñar descodificadores y otros convertidores de código en VHDL utilizando varios estilos de descripción.

Descodificador 1 de 4 líneas

La figura 1 muestra las características de un descodificador 1 de 4 líneas. Las cuatro salidas de este descodificador se activan exclusivamente, una a la vez, dependiendo del valor de la entrada. Por ejemplo, si la entrada es 2 ($a_1a_0 = 10$) únicamente la salida 2 es 1 ($y_2 = 1$, las demás salidas 0).

a_1	a_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$$y_0 = \overline{a_1} \cdot \overline{a_0}$$
$$y_1 = \overline{a_1} \cdot a_0$$
$$y_2 = a_1 \cdot \overline{a_0}$$
$$y_3 = a_1 \cdot a_0$$




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Descodificador1de4 IS
    PORT(
        a   : IN    STD_LOGIC_VECTOR(1 DOWNTO 0);
        y   : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0)
        );
END Descodificador1de4;
```

```
y(0) <= NOT a(1) AND NOT a(0);
y(1) <= NOT a(1) AND     a(0);
y(2) <=     a(1) AND NOT a(0);
y(3) <=     a(1) AND     a(0);
```

Figura 1. Tabla de verdad, funciones lógicas, cronograma, entidad y una arquitectura del descodificador 1 de 4 líneas.

También puede definirse con otras sentencias, como las indicadas en la figura 2. De esas descripciones, la más simple para expresar el funcionamiento del descodificador resulta ser la asignación selectiva (a) y luego la asignación condicional (b).

El caso (e) simplifica la descripción con sentencias IF del caso (c) especificando un valor de la salida por defecto, y luego cambiando únicamente el bit de salida afectado por la combinación de entrada. El caso (f) permite ampliar fácilmente la definición del descodificador a uno general (1 de N líneas) mediante la sentencia FOR – LOOP. Al comparar **i** con **a** en el bucle FOR debe tenerse en cuenta que la variable **i** es un entero (*integer*) y la señal **a** es un grupo de bits (*std_logic_vector*) por lo que sería necesario agregar los paquetes *arith* y *unsigned* de la biblioteca *ieee* si se empleara una arquitectura con esta sentencia.

(a) Asignación selectiva

```
WITH a SELECT
y <= "0001" WHEN "00",  -- 0
     "0010" WHEN "01",  -- 1
     "0100" WHEN "10",  -- 2
     "1000" WHEN OTHERS;-- 3
```

(b) Asignación condicional

```
y <= "0001" WHEN a = "00" ELSE
     "0010" WHEN a = "01" ELSE
     "0100" WHEN a = "10" ELSE
     "1000";    -- cuando a es 3
```

(c) Proceso con IF-ELSIF-ELSE

```
PROCESS(a)
BEGIN
    IF    a = "00" THEN y <= "0001";
    ELSIF a = "01" THEN y <= "0010";
    ELSIF a = "10" THEN y <= "0100";
    ELSE                y <= "1000";
    END IF;
END PROCESS;
```

(d) Proceso con CASE

```
PROCESS(a)
BEGIN
    CASE a IS
        WHEN "00" =>    y <= "0001";
        WHEN "01" =>    y <= "0010";
        WHEN "10" =>    y <= "0100";
        WHEN OTHERS =>  y <= "1000";
    END CASE;
END PROCESS;
```

(e) IF con inicialización por defecto

```
PROCESS(a)
BEGIN
    y <= "0000";
    IF    a = "00" THEN y(0) <= '1';
    ELSIF a = "01" THEN y(1) <= '1';
    ELSIF a = "10" THEN y(2) <= '1';
    ELSE                y(3) <= '1';
    END IF;
END PROCESS;
```

(f) Repeticiones de IF

```
PROCESS(a)
BEGIN
    y <= "0000";
    FOR i IN 0 TO 3 LOOP
        IF i = a THEN
            y(i) <= '1';
        END IF;
    END LOOP;
END PROCESS;
```

Figura 2. Descripciones VHDL del descodificador 1 de 4

Descodificador 1 de 4 líneas con habilitador

En la figura 3 se muestra un descodificador 1 de 4 líneas con habilitador. Este habilitador también puede funcionar como entrada de datos para obtener un demultiplexor o repartidor de datos. Cuando la entrada **e** es 1 el circuito funciona como en el caso anterior (la salida descodificada se coloca en 1). Cuando **e** es 0 todas las salidas se colocan en 0. En las descripciones de la figura 3 se evalúa o lista el valor del habilitador para determinar el valor de la salida. Note que esta señal tiene prioridad sobre la entrada de código.

e	a_1	a_0	y_3	y_2	y_1	y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

```
ENTITY Descodificador1de4e IS
    PORT
    (   e   : IN    STD_LOGIC;
        a   : IN    STD_LOGIC_VECTOR(1 DOWNTO 0);
        y   : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END Descodificador1de4e;
```

(a) Asignación selectiva

```
d <= e & a; -- d es un vector de 3 bits

WITH d SELECT
y <= "0001" WHEN "100", -- 0
     "0010" WHEN "101", -- 1
     "0100" WHEN "110", -- 2
     "1000" WHEN "111", -- 3
     "0000" WHEN OTHERS;-- si e es 0
```

(b) Asignación condicional

```
    y <= "0000" WHEN e = '0' ELSE
         "0001" WHEN a = "00" ELSE
         "0010" WHEN a = "01" ELSE
         "0100" WHEN a = "10" ELSE
         "1000";  -- cuando a es 3
```

(c) Proceso con IF-ELSIF-ELSE

```
PROCESS(a, e)
BEGIN
    IF e = '1' THEN
        IF    a = "00" THEN y <= "0001";
        ELSIF a = "01" THEN y <= "0010";
        ELSIF a = "10" THEN y <= "0100";
        ELSE                y <= "1000";
        END IF;
    ELSE
        y <= "0000";
    END IF;
END PROCESS;
```

(d) Proceso con CASE

```
d <= e & a; -- d es un vector de 3 bits

PROCESS(d)
BEGIN
    CASE d IS
        WHEN "100" =>   y <= "0001";
        WHEN "101" =>   y <= "0010";
        WHEN "110" =>   y <= "0100";
        WHEN "111" =>   y <= "1000";
        WHEN OTHERS =>  y <= "0000";
    END CASE;
END PROCESS;
```

(e) IF con inicialización por defecto

```
PROCESS(a, e)
BEGIN
    y <= "0000";
    IF e = '1' THEN
        IF    a = "00" THEN y(0) <= '1';
        ELSIF a = "01" THEN y(1) <= '1';
        ELSIF a = "10" THEN y(2) <= '1';
        ELSE                y(3) <= '1';
        END IF;
    END IF;
END PROCESS;
```

(f) Repeticiones de IF

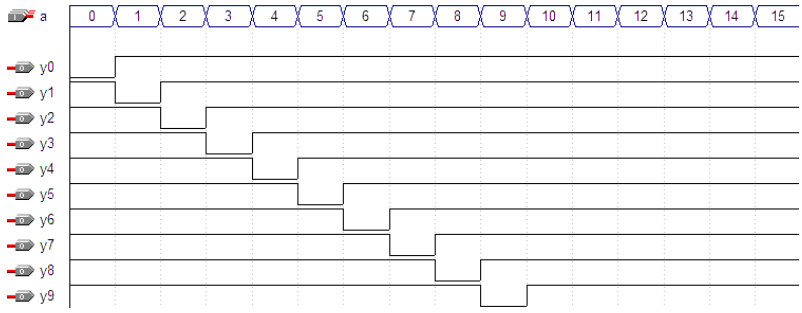
```
    PROCESS(a)
    BEGIN
        y <= "0000";
        IF e = '1' THEN
            FOR i IN 0 TO 3 LOOP
                IF i = a THEN
                    y(i) <= '1';
                END IF;
            END LOOP;
        END IF;
    END PROCESS;
```

Figura 3. Descripciones VHDL del descodificador 1 de 4 líneas con habilitador

Modelos del descodificador 74LS42

El circuito integrado 74LS42 es un descodificador que posee una entrada BCD de cuatro bits y diez salidas mutuamente exclusivas activas en 0. Cuando se presenta un código de entrada mayor que 9 todas las salidas se colocan en 1. La tabla de verdad de este circuito y un cronograma se muestran en la figura 4.

A_0	A_1	A_2	A_3	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$	$\overline{8}$	$\overline{9}$
L	L	L	L	L	H	H	H	H	H	H	H	H	H
H	L	L	L	H	L	H	H	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H	H	H
H	H	L	L	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	H	H	H	L	H	H	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
H	H	H	L	H	H	H	H	H	H	H	L	H	H
L	L	L	H	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
L	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H




```
ENTITY Descodificador7442 IS
    PORT
    (   a   : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        y   : OUT   STD_LOGIC_VECTOR(0 TO 9)
    );
END Descodificador7442;
```

Figura 4. Tabla de verdad, cronograma y entidad para el descodificador 74LS42

La descripción de este descodificador se obtiene ampliando la descripción del descodificador 1 de 4 sin habilitador: de 2 entradas a 4 entradas y de 4 salidas a 10 salidas. Dos arquitecturas, con asignación selectiva y condicional, se describen a continuación:

```
WITH a SELECT
y <= "0111111111" WHEN "0000",
     "1011111111" WHEN "0001",
     "1101111111" WHEN "0010",
     "1110111111" WHEN "0011",
     "1111011111" WHEN "0100",
     "1111101111" WHEN "0101",
     "1111110111" WHEN "0110",
     "1111111011" WHEN "0111",
     "1111111101" WHEN "1000",
     "1111111110" WHEN "1001",
     "1111111111" WHEN OTHERS;
```

```
y <= "0111111111" WHEN a = "0000" ELSE
     "1011111111" WHEN a = "0001" ELSE
     "1101111111" WHEN a = "0010" ELSE
     "1110111111" WHEN a = "0011" ELSE
     "1111011111" WHEN a = "0100" ELSE
     "1111101111" WHEN a = "0101" ELSE
     "1111110111" WHEN a = "0110" ELSE
     "1111111011" WHEN a = "0111" ELSE
     "1111111101" WHEN a = "1000" ELSE
     "1111111110" WHEN a = "1001" ELSE
     "1111111111";
```

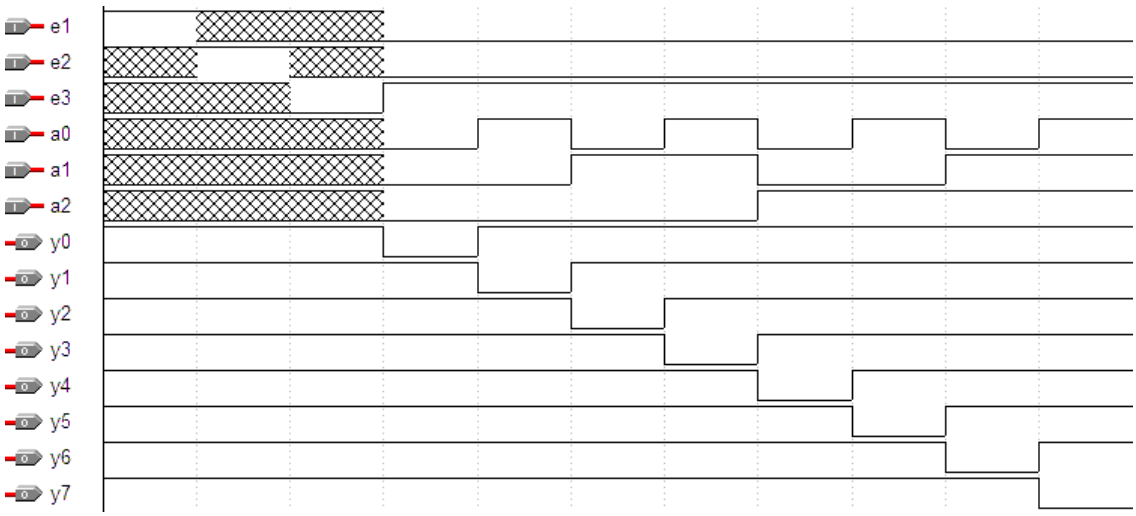
Ejercicio 1

1. Como en el ejemplo del descodificador 1 de 4, determine descripciones similares a los casos (c), (d), (e) y (f) de la figura 2 para el descodificador 7442.
2. Describa un descodificador 1 de 5 líneas con una asignación condicional.
3. Describa un descodificador 1 de 12 líneas con una asignación selectiva.
4. Describa un descodificador 1 de 64 líneas.

Modelos del descodificador 74LS138

El circuito integrado 74LS138 es un descodificador de 1 de 8 líneas con tres habilitadores. Cuando está habilitado, el circuito acepta un código binario de tres entradas (A0–A2, A2 es el MSB) y entrega salidas activas en BAJA mutuamente exclusivas (O0–O7). Dos habilitadores son activos en baja (E1, E2) y uno es activo en alta (E3). Si no se presentan las condiciones de activación todas las salidas aparecen en el nivel ALTO.

ENTRADAS						SALIDAS							
E_1	E_2	E_3	A_0	A_1	A_2	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Descodificador74138 IS
    PORT(
        a   : IN    STD_LOGIC_VECTOR(2 DOWNTO 0);
        e   : IN    STD_LOGIC_VECTOR(3 DOWNTO 1);
        y   : OUT   STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
END Descodificador74138;
```

Figura 5. Tabla de verdad, cronograma y entidad par el descodificador 74LS138

La descripción de este descodificador se obtiene ampliando la descripción del descodificador 1 de 4 con habilitador. Por ejemplo, en la asignación selectiva se agrupan los bits de los habilitadores y de las entradas para seleccionar el valor de la salida:

```
d <= e & a; -- d es un vector de 6 bits

WITH d SELECT
y <= "01111111" WHEN "001000",
     "10111111" WHEN "001100",
     "11011111" WHEN "001010",
     "11101111" WHEN "001110",
     "11110111" WHEN "001001",
     "11111011" WHEN "001101",
     "11111101" WHEN "001011",
     "11111110" WHEN "001111",
     "11111111" WHEN OTHERS;
```

En la asignación condicional primero se evalúa el estado de los habilitadores y después el código de entrada:

```
y <= "11111111" WHEN e(1) = '1' OR e(2) = '1' OR e(3) = '0' ELSE
     "01111111" WHEN a = "000" ELSE
     "10111111" WHEN a = "100" ELSE
     "11011111" WHEN a = "010" ELSE
     "11101111" WHEN a = "110" ELSE
     "11110111" WHEN a = "001" ELSE
     "11111011" WHEN a = "101" ELSE
     "11111101" WHEN a = "011" ELSE
     "11111110";
```

Similarmente, en un proceso con IF – ELSIF – ELSE se evalúan los habilitadores y luego el código de entrada:

```
PROCESS(a, e)
BEGIN
    IF e = "001" THEN
        IF    a = "000" THEN y <= "01111111";
        ELSIF a = "100" THEN y <= "10111111";
        ELSIF a = "010" THEN y <= "11011111";
        ELSIF a = "110" THEN y <= "11101111";
        ELSIF a = "001" THEN y <= "11110111";
        ELSIF a = "101" THEN y <= "11111011";
        ELSIF a = "011" THEN y <= "11111101";
        ELSE                 y <= "11111110";
        END IF;
    ELSE
        y <= "11111111";
    END IF;
END PROCESS;
```

En el lazo iterativo con FOR – LOOP se utiliza una señal auxiliar **n** para reordenar los bits del código de entrada (el bit a(2) es el MSB). En el proceso se asigna a la salida el valor del circuito cuando está deshabilitado, luego se evalúa la combinación de habilitadores y después se evalúa la combinación del código de entrada:

```
n <= a(2) & a(1) & a(0);

PROCESS(a)
BEGIN
    y <= "11111111";
    IF e = "001" THEN
        FOR i IN 0 TO 7 LOOP
            IF i = n THEN
                y(i) <= '0';
            END IF;
        END LOOP;
    END IF;
END PROCESS;
```

Ejercicio 2

1. Describa el circuito 74LS138 con un proceso usando CASE. Simule el circuito para obtener la simulación de la figura 5.

2. El circuito integrado 74LS139 contiene dos descodificadores/demultiplexores 1 de 4 independientes. Cada dispositivo acepta dos entradas y provee cuatro salidas mutuamente exclusivas activas en 0. Cada bloque cuenta con un habilitador activo en 0. Cada bloque cumple con la siguiente tabla de verdad:

e	i_1	i_0	y_0	y_1	y_2	y_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Considerando la entidad indicada, realice un modelo VHDL para este descodificador.

```
ENTITY Descodificador74139 IS
    PORT
    (   -- descodificador A
        ea  : IN    STD_LOGIC;
        a   : IN    STD_LOGIC_VECTOR(1 DOWNTTO 0);
        ya  : OUT   STD_LOGIC_VECTOR(0 TO 3);
        -- descodificador B
        eb  : IN    STD_LOGIC;
        b   : IN    STD_LOGIC_VECTOR(1 DOWNTTO 0);
        yb  : OUT   STD_LOGIC_VECTOR(0 TO 3)
    );
END Descodificador74139;
```

Descodificador 1 de 32 líneas utilizando 74138

Un descodificador 1 de 32 líneas se realiza fácilmente extendiendo la descripción de un descodificador 1 de 8 líneas. En este ejemplo se mostrará cómo hacerlo utilizando componentes, según el circuito esquemático de la figura 6. El circuito opera de acuerdo al cronograma de la figura 7. Una descripción estructural de este circuito se indica en la página siguiente.

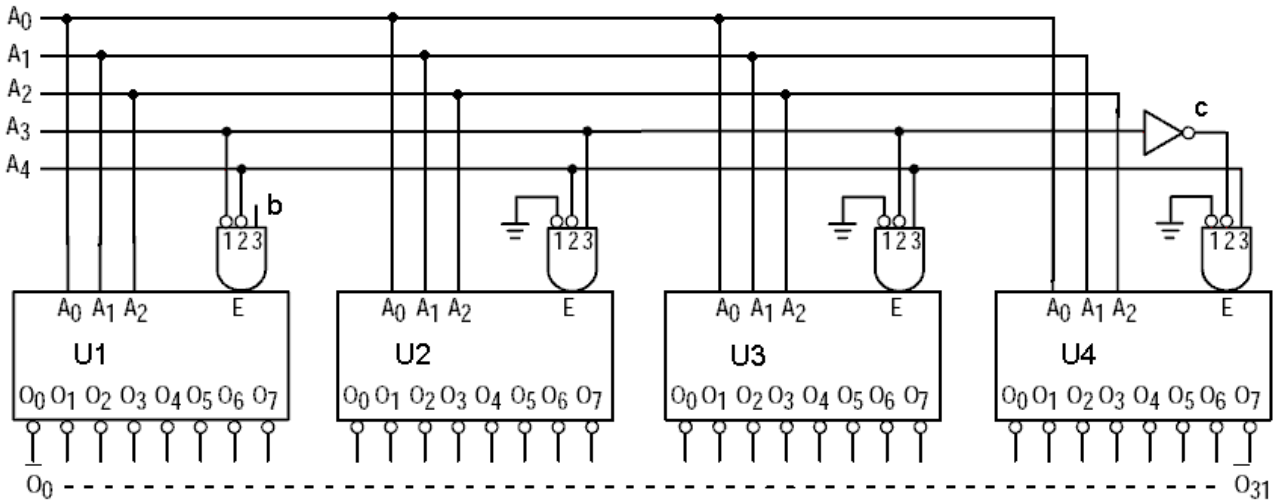


Figura 6. Circuito esquemático de un descodificador 1 de 32 líneas

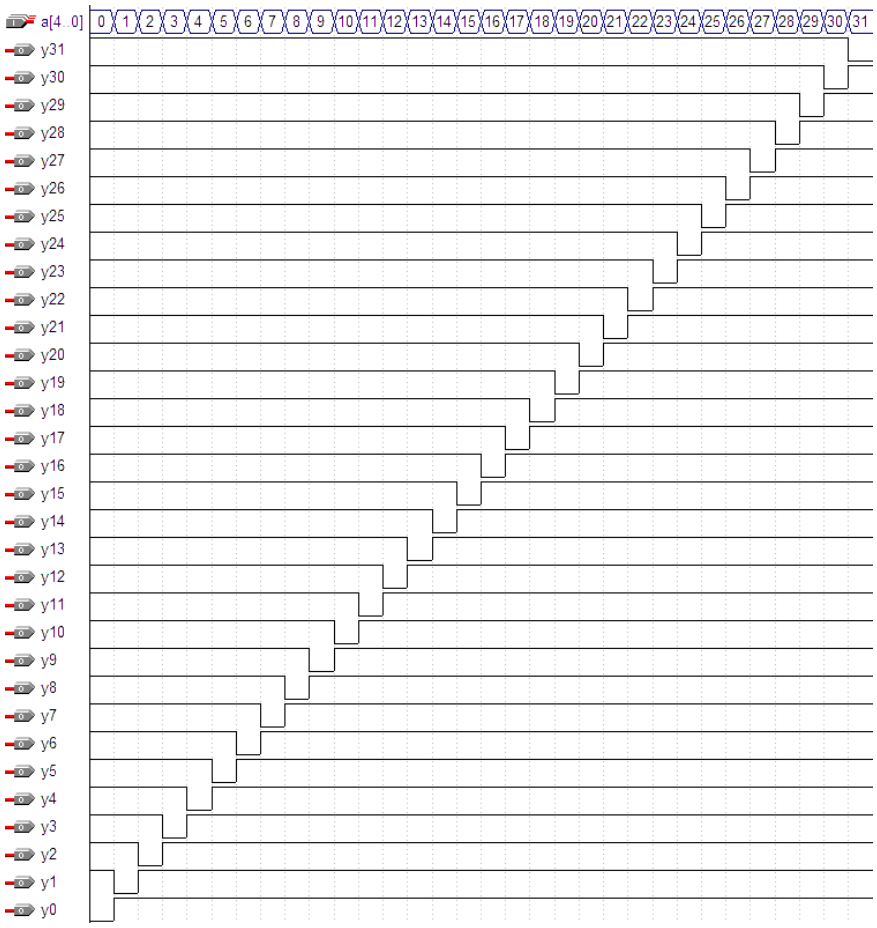


Figura 7. Cronograma del descodificador 1 de 32 líneas

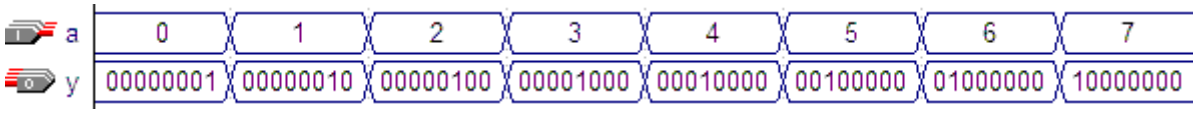
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Descodificador1de32 IS
    PORT
    (   a   : IN    STD_LOGIC_VECTOR(0 TO 4);
        y   : OUT   STD_LOGIC_VECTOR(0 TO 31)
    );
END Descodificador1de32;

ARCHITECTURE estructural OF Descodificador1de32 IS

    COMPONENT Descodificador74138 IS
    PORT
    (   e   : IN    STD_LOGIC_VECTOR(1 TO 3);
        a   : IN    STD_LOGIC_VECTOR(0 TO 2);
        y   : OUT   STD_LOGIC_VECTOR(0 TO 7)
    );
    END COMPONENT;

    SIGNAL  e1, e2,
            e3, e4,
            d:      STD_LOGIC_VECTOR(1 TO 3);
    SIGNAL b, c: STD_LOGIC;

BEGIN

    d <= a(0) & a(1) & a(2);
    e1 <= a(3) & a(4) & '1';
    e2 <= '0' & a(4) & a(3);
    e3 <= '0' & a(3) & a(4);
    e4 <= '0' & (NOT a(3)) & a(4);

    U1: Descodificador74138
        PORT MAP (e1, d, y(0 TO 7));

    U2: Descodificador74138
        PORT MAP (e2, d, y(8 TO 15));

    U3: Descodificador74138
        PORT MAP (e3, d, y(16 TO 23));

    U4: Descodificador74138
        PORT MAP (e4, d, y(24 TO 31));

END estructural;
```

Descodificador 1 de N líneas

A continuación se presenta una descripción de un descodificador 1 de N líneas.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

ENTITY Descodificador1deNe IS
    GENERIC(NL: INTEGER:= 8;    -- número de líneas de salida
            NE: INTEGER:= 3);   -- número de bits de código
PORT
    (   a   : IN    STD_LOGIC_VECTOR(NE - 1 DOWNTO 0);
        e   : IN    STD_LOGIC;
        y   : OUT   STD_LOGIC_VECTOR(NL - 1 DOWNTO 0)
    );
END Descodificador1deNe;

ARCHITECTURE ForLoop OF Descodificador1deNe IS
BEGIN

    PROCESS(a)
    BEGIN
        y <= (OTHERS => '0');
        IF e = '1' THEN
            FOR i IN 0 TO NL - 1 LOOP
                IF i = a THEN
                    y(i) <= '1';
                END IF;
            END LOOP;
        END IF;
    END PROCESS;

END ForLoop;
```

El valor de N se define en GENERIC como NL con un valor igual a 8 por defecto. El número de bits del código de entrada es NE al que se la ha asignado 3 por defecto (3 entradas de código, 8 salidas descodificadas). En la arquitectura, el lazo FOR barre todos los valores de una variable **i** desde 0 hasta NL – 1. En cada iteración, se comprueba si el valor de **i** coincide con el valor actual de **a**. Si es así, la línea de salida correspondiente se activa en 1, sino se desactiva (en 0). Note que antes de ingresar a la iteración todos los bits de la salida se colocan a 0 por defecto con la sentencia ***y <= (OTHERS => '0')***. Una simulación muestra el caso para NE = 3 y NL = 8:

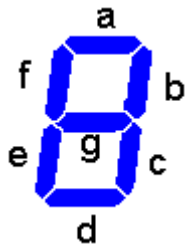


En el siguiente ejemplo se crean dos descodificadores (1 de 5 líneas y 1 de 12 líneas) utilizando el componente del descodificador 1 de N líneas. Note la definición de las entradas y la especificación particular de los valores de NL y NE para cada caso.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Descodificador1de5y1de12 IS
    PORT
    (   -- descodificador 1 de 5 líneas
        a   : IN    STD_LOGIC_VECTOR(2 DOWNTO 0);
        ea  : IN    STD_LOGIC;
        ya  : OUT   STD_LOGIC_VECTOR(4 DOWNTO 0);
        -- descodificador 1 de 12 líneas
        b   : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        eb  : IN    STD_LOGIC;
        yb  : OUT   STD_LOGIC_VECTOR(11 DOWNTO 0));
END Descodificador1de5y1de12;

ARCHITECTURE estructural OF Descodificador1de5y1de12 IS
    COMPONENT Descodificador1deNe IS
    GENERIC(NL: INTEGER:= 8;
            NE: INTEGER:= 3);
    PORT(   a   : IN    STD_LOGIC_VECTOR(NE - 1 DOWNTO 0);
            e   : IN    STD_LOGIC;
            y   : OUT   STD_LOGIC_VECTOR(NL - 1 DOWNTO 0));
    END COMPONENT;

BEGIN

    UA: Descodificador1deNe
    GENERIC MAP(NL => 7, NE => 3)
    PORT MAP(a, ea, ya);

    UB: Descodificador1deNe
    GENERIC MAP(NL => 16, NE => 4)
    PORT MAP(b, eb, yb);

END estructural;
```

Descodificador de 7 segmentos

Un descodificador de siete segmentos recibe un número en código BCD y entrega otro código para un visualizador de 7 segmentos. A continuación se muestran dos descodificadores de este tipo, uno con salidas activas en baja y otro con salidas activas en alta, para visualizadores de ánodo y cátodo común, respectivamente.



El circuito sirve para mostrar los números de la siguiente manera:

En la entidad, la salida q(6) corresponde con el segmento a y q(0) con el segmento g. En la arquitectura el circuito descodifica las entradas 0 a 9 (BCD) y activa las señales correspondientes a los segmentos a..g. Cuando la entrada es mayor que 9 las salidas hacen que los segmentos permanezcan apagados.

```
ENTITY Dec7Segmentos IS
    PORT(
        d   : IN    BIT_VECTOR(3 DOWNTO 0);
        q   : OUT   BIT_VECTOR(6 DOWNTO 0));
END Dec7Segmentos;
```

Salidas activas en baja

```
WITH d SELECT
    q <= "0000001"   WHEN "0000",
         "1001111"   WHEN "0001",
         "0010010"   WHEN "0010",
         "0000110"   WHEN "0011",
         "1001100"   WHEN "0100",
         "0100100"   WHEN "0101",
         "0100000"   WHEN "0110",
         "0001111"   WHEN "0111",
         "0000000"   WHEN "1000",
         "0000100"   WHEN "1001",
         "1111111"   WHEN OTHERS;
```

Salidas activas en alta

```
WITH d SELECT
    q <= "1111110" WHEN "0000", -- 0
         "0110000" WHEN "0001", -- 1
         "1101101" WHEN "0010", -- 2
         "1111001" WHEN "0011", -- 3
         "0110011" WHEN "0100", -- 4
         "1011011" WHEN "0101", -- 5
         "1011111" WHEN "0110", -- 6
         "1110000" WHEN "0111", -- 7
         "1111111" WHEN "1000", -- 8
         "1111011" WHEN "1001", -- 9
         "0000000" WHEN OTHERS;
```

Ejercicio 3

Describa un circuito para incluir la codificación en visualizadores de siete segmentos de los números del 10 al 15 según el patrón siguiente.

Conclusiones

1. Este módulo ha cubierto varios tipos de descodificadores en VHDL con diferentes estilos de descripción.

2. Los descodificadores de pocas líneas se realizan fácilmente con asignaciones selectivas y condicionales.

3. Las sentencias IF combinadas con FOR – LOOP permiten crear descodificadores con habilitadores y cualquier número de líneas.

4. En el próximo módulo se presentarán los multiplexores, circuitos ampliamente utilizados en el diseño digital.