Curso a distancia:

INTRODUCCIÓN AL DISEÑO LÓGICO PROGRAMABLE CON VHDL

Capítulo 3

Codificadores

Codificadores binarios y codificadores de prioridad.
Codificadores de 3 a 2 líneas y de 4 a dos líneas.
Detector de dirección cardinal. Detector de tecla pulsada. Medidor de nivel de agua.
Modelos de los codificadores 74147 y 74148.
Codificador de 16 a 4 líneas utilizando codificadores 74148.
Codificador de prioridad de N a M líneas.

Ing. Arturo J. Miguel de Priego Paz Soldán
www.tourdigital.net
Chincha – Perú, 23 de octubre de 2008

Introducción

Un codificador analiza un conjunto de líneas de entradas para determinar un número que identifica a las líneas activadas. Los codificadores más empleados son los llamados codificadores de prioridad, donde la salida muestra el número de la entrada activada con mayor prioridad. Estos codificadores se emplean típicamente en circuitos detectores de teclas pulsadas, medidores de niveles, y detectores de líneas de interrupción en un sistema basado en microprocesador. Otro tipo de codificadores es el codificador binario, donde las señales de entrada solamente se activan una a la vez. Ellos se utilizan, por ejemplo, en circuitos de detección de posición, como brújulas o perillas selectoras.

En este capítulo se presentan varias descripciones VHDL de distintos codificadores. Se describe un codificador binario de 4 entradas, un codificador de prioridad de 3 a 2 líneas y otro con prioridad de 4 a 2 líneas. Luego se presentan las descripciones de dos codificadores de prioridad de circuito integrado, el 74LS147 y el 74LS148. Después se desarrolla un modelo para un codificador general parametrizable de N a M líneas.

Objetivos Instruccionales

Al finalizar este capítulo, el lector estará en capacidad de:

1. Describir en VHDL codificadores binarios y codificadores de prioridad con asignaciones condicionales y selectivas, y con procesos usando IF – ELSIF – ELSE y CASE.

2. Describir codificadores generales de N a M líneas utilizando procesos con LOOP – FOR.

Codificador binario de 3 entradas

La tabla de verdad de un codificador binario de 3 entradas se muestra en la figura 1. Las señales a, b y c se asocian a las líneas de entrada 3, 2 y 1 respectivamente. El código de salida corresponde a la línea activada. Si ninguna o más de una entrada está activada la salida se coloca en 0. De acuerdo a esto, las combinaciones de las filas T2, T3 y T5 proveen salidas válidas, mientras que para el resto de combinaciones la salida es cero. La tabla de verdad simplificada, que aparece en el lado derecho de la figura, provee información concisa de este funcionamiento.

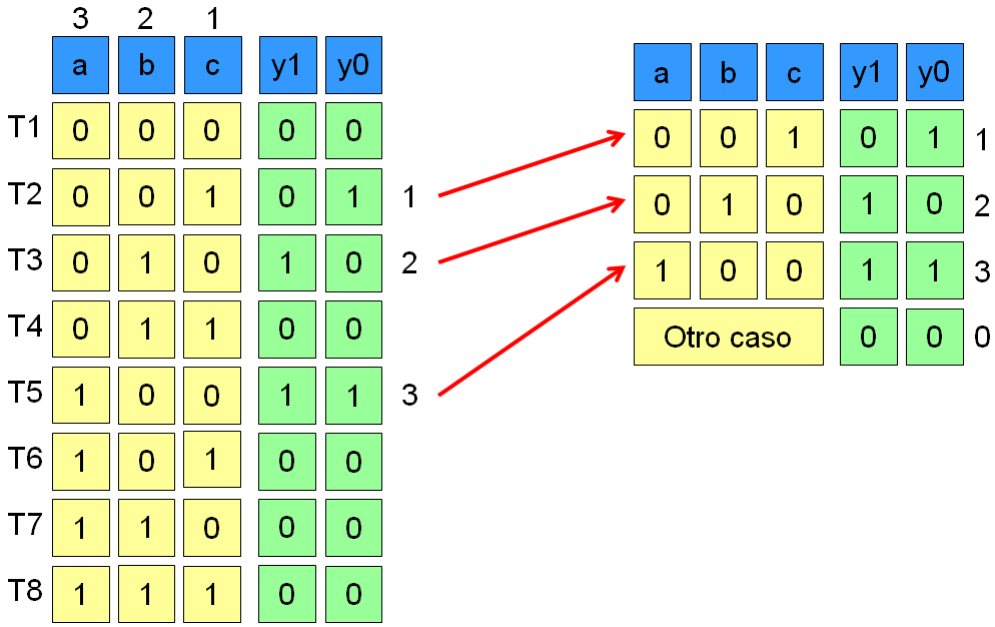


Figura 1. Tablas de verdad de un codificador binario de tres entradas

El cronograma de este circuito se muestra en la figura 2. En T2 la señal **c** es 1 y la salida **y** es 01 correspondiente al número de canal de la señal **c**. Similarmente, en T3 la salida es el número binario 10, ó 2 en decimal, porque la señal **b**, conectada a la línea 2, está activa. En T5 la señal **a** está activa haciendo que la salida sea el número binario 11, ó 3 en decimal. En todos los demás casos la salida es cero. En T1 ninguna entrada está activada, y en T4, T6, T7 y T8 más de una señal lo está.

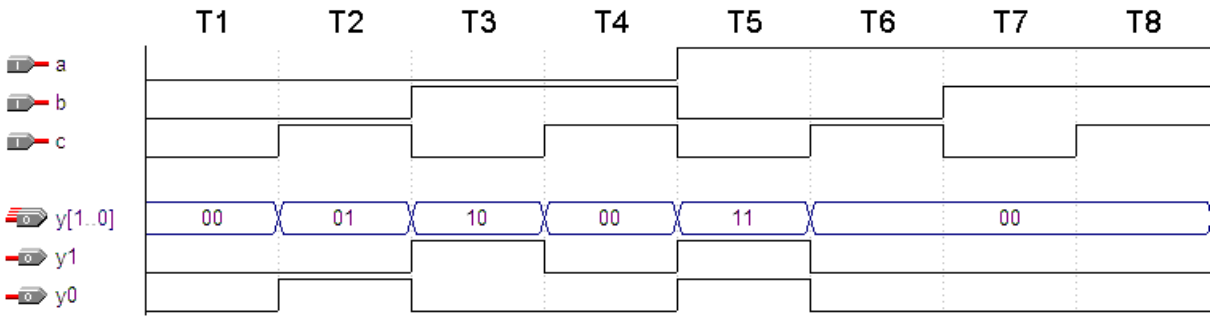


Figura 2. Cronograma del codificador binario de tres entradas

La entidad de este codificador puede definirse de la siguiente manera:

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

ENTITY Codificador3a2 IS
    PORT
    (   a, b, c : IN    STD_LOGIC;
        y       : OUT   STD_LOGIC_VECTOR(1 DOWNTO 0)
    );
END Codificador3a2;
```

Utilizando una asignación selectiva partiendo de la tabla de verdad simplificada, agrupando previamente los bits de entrada en un arreglo interno de tres bits, resulta la arquitectura:

```
ARCHITECTURE selectiva OF Codificador3a2 IS
    SIGNAL d: STD_LOGIC_VECTOR(1 TO 3);
BEGIN

    d <= a & b & c;

    WITH d SELECT
    y <= "01" WHEN "001",
         "10" WHEN "010",
         "11" WHEN "100",
         "00" WHEN OTHERS;

END selectiva;
```

Con asignación condicional, una descripción equivalente es:

```
ARCHITECTURE condicional OF Codificador3a2 IS
BEGIN

   y <= "11" WHEN a = '1' AND b = '0' AND c = '0' ELSE
        "10" WHEN a = '0' AND b = '1' AND c = '0' ELSE
        "01" WHEN a = '0' AND b = '0' AND c = '1' ELSE
        "00";

END condicional;
```

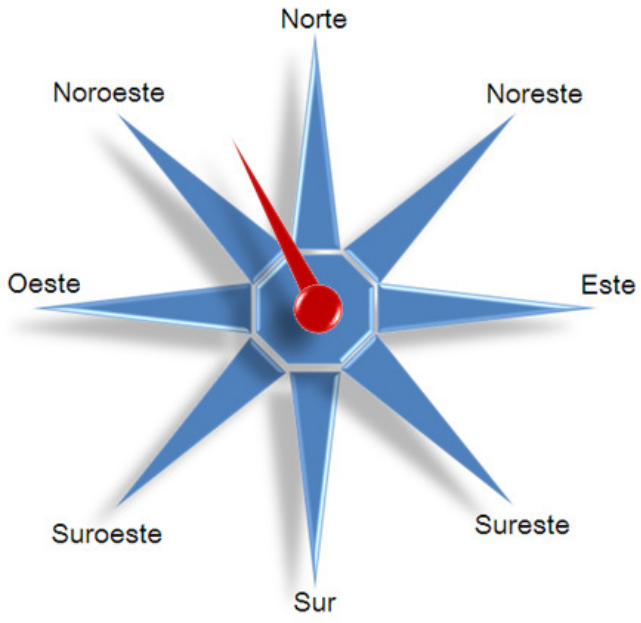
Este codificador binario es útil en casos donde solamente una entrada puede estar activada. En otros casos se suele utilizar codificadores de prioridad.

Ejercicio 1

1. Describa el codificador de la figura 1 con un proceso usando IF – ELSIF – ELSE.
2. Describa el mismo codificador utilizando un proceso con CASE.
3. De las cuatro arquitecturas ¿Cuál es la más sencilla para describir codificadores binarios?

Ejercicio 2

En la figura 3 se muestra una brújula operada mecánicamente de modo tal que solamente una de las ocho líneas de direcciones puede estar en contacto con la aguja en un momento dado. Las ocho direcciones se conectan a un circuito digital. Cuando una dirección hace contacto con la aguja la señal se activa en el nivel lógico 1. Cuando no hay contacto la señal está en el nivel lógico 0. Por ejemplo, si la aguja está apuntando hacia Noreste, la salida deberá mostrar el código 001.



Entrada	**:**	**Salida**
Norte	:	000
Noreste	:	001
Este	:	010
Sureste	:	011
Sur	:	100
Suroeste	:	101
Oeste	:	110
Noroeste	:	111

Figura 3. Brújula de ocho direcciones

Diseñe en VHDL, con la arquitectura más simple posible, un circuito digital para mostrar la dirección de la aguja. En la entidad, identifique las entradas con los nombres que aparecen en la figura. La salida debe ser de tipo std_logic_vector.

Codificador con prioridad de 3 a 2 líneas

En el circuito previo existe la posibilidad de que más de una entrada se active. En tal caso la salida se fijaba a cero. Si se define un sistema de prioridades es posible asignar una salida correspondiente a la línea con prioridad más alta en caso de que más de una línea se active.

El circuito esquemático de la figura 4 es un codificador con prioridad de tres entradas y dos salidas. Cuando una línea de entrada se activa la salida muestra su número de línea en binario. Si más de una entrada se activa al mismo tiempo, la salida codifica la entrada con mayor prioridad. En este ejemplo, la entrada de mayor prioridad es **a**, luego **b** y después sigue **c** con la menor prioridad.

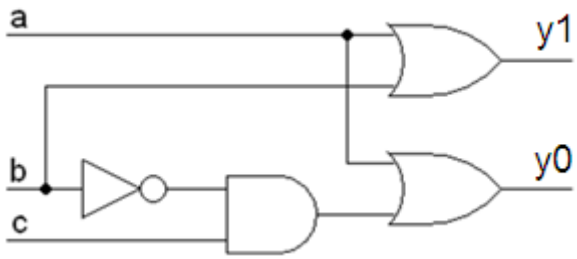


Figura 4. Circuito lógico de un codificador con prioridad de 3 a 2 líneas

En la figura 5 se indican dos tablas de verdad de este circuito. La primera tabla muestra todas las combinaciones posibles de las tres entradas y los valores respectivos de la salida. La segunda tabla es una versión simplificada de la primera que utiliza valores de no importa para ciertas combinaciones de entradas y es la que típicamente se utiliza para describir tablas de verdad de codificadores con prioridad.

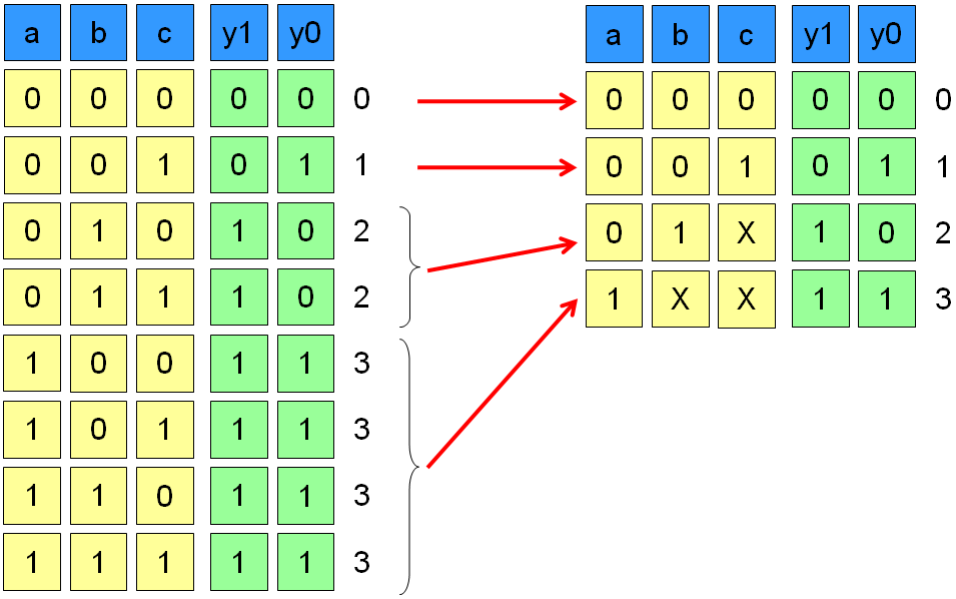


Figura 5. Tablas de verdad de un codificador con prioridad de 3 a 2 líneas

En el cronograma de la figura 6 se muestran 12 casos de combinaciones de entradas. Los primeros 8 corresponden con la primera tabla de verdad, mientras que los cuatro siguientes reflejan la tabla simplificada. En T1 todas las entradas están en cero, por lo tanto la salida es 00 indicando que

ninguna entrada está activa. En T2 se activa únicamente la entrada **c** y la salida es 01 indicando que la entrada del canal 1 está activa. En T3 y T4 se activa la segunda entrada, **b**, por lo tanto la salida es 10, sin importar el valor de **c**. Desde T5 hasta T8 la entrada **a**, de mayor prioridad, está activada causando que la salida sea 11, sin importar los valores de b y c. En T9 ocurre lo mismo que en T1, en T10 pasa lo que sucede en T2, en T11 se cubren los casos de T3 y T4, y en T12 se presenta una combinación equivalente a cualquiera de los tiempos T5, T6, T7 o T8.

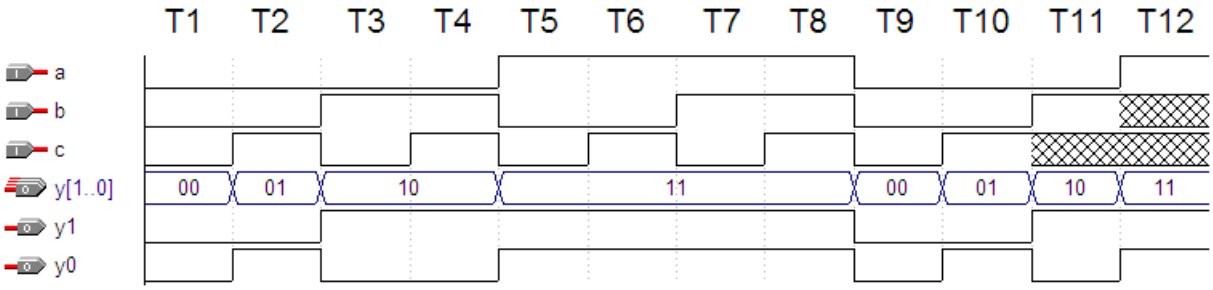


Figura 6. Cronograma de un codificador con prioridad de 3 a 2 líneas

A continuación se indican cuatro arquitecturas diferentes acompañadas de pseudocódigos. Todas las descripciones son equivalentes, definen el mismo circuito codificador de la figura 4. La entidad VHDL es igual al codificador binario previo.

Asignación selectiva

De acuerdo a la combinación abc
Asignar a **y**
00 cuando sea 000,
01 cuando sea 001
10 cuando sea 010 ó 011
11 en otro caso [100, 101, 110 ó 111]

```
ARCHITECTURE selectiva OF Codificador3a2 IS
    SIGNAL d: STD_LOGIC_VECTOR(1 TO 3);
BEGIN

    d <= a & b & c;

    WITH d SELECT
    y <= "00" WHEN "000",
         "01" WHEN "001",
         "10" WHEN "010" | "011",
         "11" WHEN OTHERS;

END selectiva;
```

Asignación condicional

Esta asignación determina automáticamente una prioridad de asignación.

Hacer **y** igual a
11 cuando a sea 1, sino
10 cuando b sea 1, sino
01 cuando c sea 1, sino
00

```
ARCHITECTURE condicional OF Codificador3a2 IS
BEGIN

    y <= "11" WHEN a = '1' ELSE
         "10" WHEN b = '1' ELSE
         "01" WHEN c = '1' ELSE
         "00";

END condicional;
```

Proceso con IF – ELSIF -ELSE

Es similar a la asignación condicional.

Si a es 1 entonces y = 11
Sino, si b es 1 entonces y = 10
Sino, si c es 1 entonces y = 01
Sino y = 00

```
ARCHITECTURE ifelsifelse OF Codificador3a2 IS
BEGIN

PROCESS (a, b, c)
BEGIN
    IF a = '1' THEN     y <= "11";
    ELSIF b = '1' THEN  y <= "10";
    ELSIF c = '1' THEN  y <= "01";
    ELSE                y <= "00";
    END IF;
END PROCESS;

END ifelsifelse;
```

Proceso con CASE

Es similar a la asignación selectiva.

Según la combinación abc

Cuando sea 000:	y = 00
Cuando sea 001:	y = 01
Cuando sea 010:	y = 10
Cuando sea 011:	y = 10
Cuando sea otro valor:	y = 11

```
ARCHITECTURE casos OF Codificador3a2 IS
    SIGNAL d: STD_LOGIC_VECTOR(1 TO 3);
BEGIN

    d <= a & b & c;

    PROCESS(d)
    BEGIN
        CASE d IS
            WHEN "000" =>   y <= "00";
            WHEN "001" =>   y <= "01";
            WHEN "010" =>   y <= "10";
            WHEN "011" =>   y <= "10";
            WHEN OTHERS =>  y <= "11";
        END CASE;
    END PROCESS;

END casos;
```

Ejercicio 3

1. De las cuatro arquitecturas ¿Cuál es la más sencilla para describir codificadores con prioridad?

2. Describa completamente este codificador con asignación condicional y obtenga la simulación de la figura 6.

Codificador de prioridad de 4 a 2 líneas

La tabla de verdad y un cronograma de un codificador de 4 a 2 líneas se indica en la figura 7. En este codificador los canales se han enumerado desde 0 hasta 3. Las entradas son i3, i2, i1, i0 y sus códigos de activación son 3, 2, 1 y 0, respectivamente. La entrada i3 es la de mayor prioridad. La señal de salida **v** es cero cuando todas las entradas están en 0, y 1 si al menos una entrada está activada. Esta salida sirve para indicar que el número codificado es válido.

i3	i2	i1	i0	f1	f0	v	
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	X	0	1	1	1
0	1	X	X	1	0	1	2
1	X	X	X	1	1	1	3

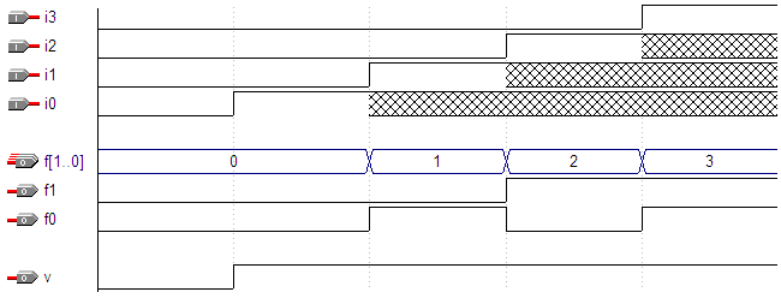


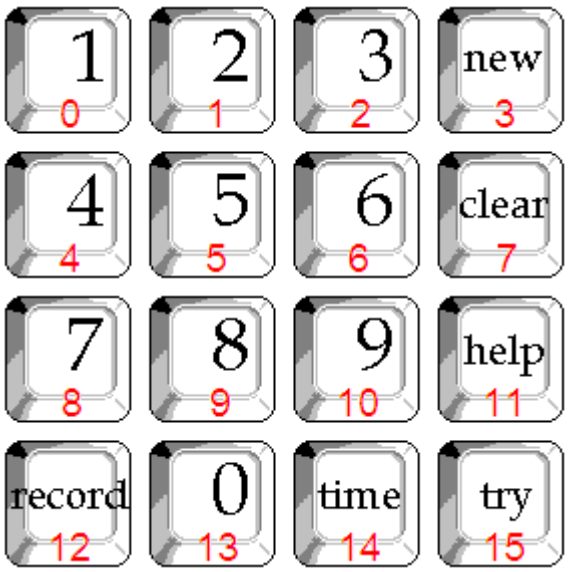
Figura 7. Tabla de verdad y cronograma de un codificador con prioridad de 4 a 2 líneas.

Una descripción VHDL de este circuito se muestra a continuación. En la entidad las señales de entrada de datos y la salida codificada están definidos como vectores. En la arquitectura las asignaciones son condicionales porque resultan la manera más práctica para describir este tipo de codificadores.

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

ENTITY Codificador4a2 IS
    PORT
    (   i   : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        f   : OUT   STD_LOGIC_VECTOR(1 DOWNTO 0);
        v   : OUT   STD_LOGIC
    );
END Codificador4a2;

ARCHITECTURE condicional OF Codificador4a2 IS

BEGIN

    f <=    "11" WHEN i(3) = '1' ELSE
            "10" WHEN i(2) = '1' ELSE
            "01" WHEN i(1) = '1' ELSE
            "00";

    v <= '0' WHEN i = "0000" ELSE '1';

END condicional;
```

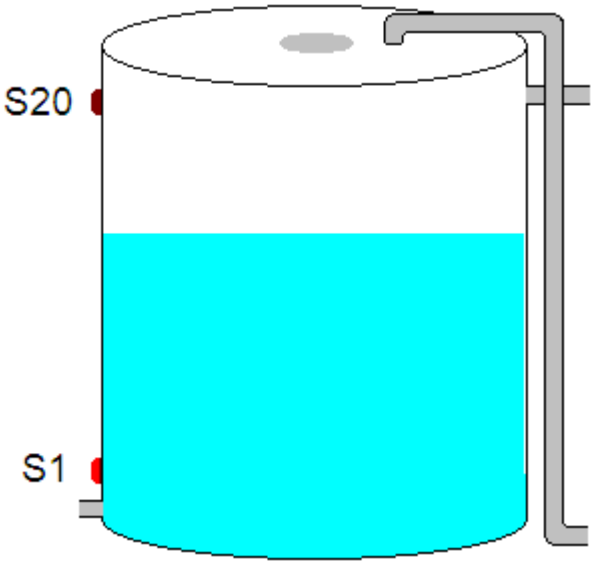
Ejercicio 4

En un teclado de 16 teclas se han ubicado los dígitos como se muestra en la figura. Cuando se pulsa una tecla se activa una señal correspondiente en el estado 1 lógico. Realice una descripción VHDL para codificar la tecla pulsada con el número identificador de la tecla (en rojo). Las salidas deben estar activas en alta. Por ejemplo si se pulsa la tecla 0, la salida será 0000 (se encuentra en la posición o línea 0); si se pulsa la tecla 7 la salida será 1000 (se encuentra en la posición o línea 8). Si dos o más teclas están presionadas la salida codificará la tecla de mayor prioridad, siendo la tecla en la posición 0 la de mayor prioridad y la tecla en la posición 15 la de menor prioridad. Cuando ninguna tecla esté presionada el circuito lo anunciará en una salida llamada gs activa en baja. Si hay tecla presionada entonces esta señal se coloca en 1 lógico.



Ejercicio 5

En un tanque de agua se han colocado 20 sensores (S1..S20) que se activan en 1 lógico con la presión de agua. Diseñe un circuito para mostrar el nivel de agua en números BCD. Por ejemplo, si el nivel de agua alcanza al sensor 15 la salida será 0001 0101, correspondiendo los primeros cuatro bits para el número 1 y los últimos cuatro bits para el número 5; igualmente, si el nivel de agua llega hasta el sensor 9 entonces la salida será 0000 1001.



Un modelo del circuito integrado 74LS147

El circuito integrado 74LS147 es un codificador con prioridad de 9 entradas que se activan en el nivel lógico bajo (0). La salida, de cuatro bits, codifica la entrada activa de mayor prioridad proporcionando el número de línea en complemento a 1. Si ninguna entrada está activada (si todas las entradas están en 1), la salida es 1111 (0000 en complemento a 1).

La tabla de la figura 8 detalla el funcionamiento de este circuito. La línea 9 es la de mayor prioridad y la línea 1 la de menor prioridad. La salida muestra un número (conformado por las líneas A, B, C, D) indicando la línea activada de mayor prioridad. La salida D es el bit más significativo (MSB) y la salida A es el bit menos significativo (LSB).

ENTRADAS									SALIDAS			
1	2	3	4	5	6	7	8	9	D	C	B	A
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	L	H	H	H
X	X	X	X	X	X	L	H	H	H	L	L	L
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	L	H	L
X	X	X	L	H	H	H	H	H	H	L	H	H
X	X	L	H	H	H	H	H	H	H	H	L	L
X	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

Figura 8. Tabla de verdad del circuito 74LS147

Un cronograma equivalente a la tabla de verdad se muestra en la figura 9. En T1 todas la entradas están en 1, por lo tanto la salida es 1111 (0000 en complemento a 1). En T2 se activa la línea 9, representada por la señal i9. La salida en ese momento es 0110 (1001 en complemento a 1). En T2 las señales 1 a 8 no influyen en el valor de salida, porque i9 tiene la mayor prioridad. En T10 la entrada i1 está activada, por lo que la salida es 0001 en complemento a 1 (1110).

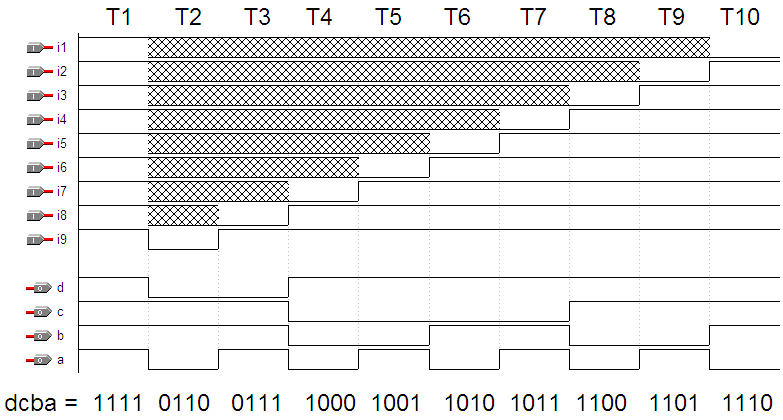


Figura 9. Un cronograma para el circuito 74LS147

En la siguiente descripción VHDL, las entradas se agrupan en una señal i de tipo std_logic_vector con índices de 1 a 9. Internamente, se declara una señal n para asignar los valores de la salida como patrones de bits, y luego se separa cada bit de salida para a, b, c y d.

La entidad VHDL queda definida así:

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

ENTITY Codificador74147 IS
    PORT
    (   i           : IN    STD_LOGIC_VECTOR(1 TO 9);
        a, b, c, d  : OUT   STD_LOGIC);
END Codificador74147;
```

El pseudocódigo y la arquitectura quedan expresados de la siguiente manera:

Hacer n igual a
- 0110 si la línea 9 es 0 sino
- 0111 si la línea 8 es 0 sino
- 1000 si la línea 7 es 0 sino
- 1001 si la línea 6 es 0 sino
- 1010 si la línea 5 es 0 sino
- 1011 si la línea 4 es 0 sino
- 1100 si la línea 3 es 0 sino
- 1101 si la línea 2 es 0 sino
- 1110 si la línea 1 es 0 sino
- 1111

Hacer a = n(0)
Hacer b = n(1)
Hacer c = n(2)
Hacer d = n(3)

```
ARCHITECTURE condicional OF Codificador74147 IS
    SIGNAL  n   : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN

    n <= "0110" WHEN i(9) = '0' ELSE
         "0111" WHEN i(8) = '0' ELSE
         "1000" WHEN i(7) = '0' ELSE
         "1001" WHEN i(6) = '0' ELSE
         "1010" WHEN i(5) = '0' ELSE
         "1011" WHEN i(4) = '0' ELSE
         "1100" WHEN i(3) = '0' ELSE
         "1101" WHEN i(2) = '0' ELSE
         "1110" WHEN i(1) = '0' ELSE
         "1111";

a <= n(0);
b <= n(1);
c <= n(2);
d <= n(3);

END condicional;
```

Una forma más compacta para el modelo VHDL del codificador 74147 utiliza una salida agrupada, tal como se muestra a continuación:

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

ENTITY Codificador74147e IS
    PORT
    (   i   : IN    STD_LOGIC_VECTOR(1 TO 9);
        y   : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0));
END Codificador74147e;

 ARCHITECTURE condicional OF Codificador74147e IS
 BEGIN

     y <= "0110" WHEN i(9) = '0' ELSE
          "0111" WHEN i(8) = '0' ELSE
          "1000" WHEN i(7) = '0' ELSE
          "1001" WHEN i(6) = '0' ELSE
          "1010" WHEN i(5) = '0' ELSE
          "1011" WHEN i(4) = '0' ELSE
          "1100" WHEN i(3) = '0' ELSE
          "1101" WHEN i(2) = '0' ELSE
          "1110" WHEN i(1) = '0' ELSE
          "1111";

 END condicional;
```

Un modelo del circuito integrado 74LS148

El circuito integrado 74LS148 es otro codificador con prioridad. Posee 8 líneas de activación, un habilitador de entradas, una salida de codificación de tres bits, un habilitador de salidas y una salida de señalización de grupo. Todas las entradas y salidas se activan en el nivel lógico bajo (0). La tabla de verdad para este circuito se muestra en las figura 10.

ENTRADAS									SALIDAS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

Figura 10. Tabla de verdad para el circuito integrado 74LS148.

Este circuito codifica las ocho líneas de datos (indizadas de 0 a 7) en tres líneas binarias. Posee señales para expansión de la codificación EI (*Enable Input*) y EO (*Enable Output*). EI es el habilitador de entrada. EO es el habilitador de salida: se activa cuando el chip está habilitado y ninguna línea está activa. GS es señalizador de grupo: se activa cuando el chip está habilitado y al menos una línea está activa.

La línea 7 es la de mayor prioridad y la línea 0 es la de menor prioridad. Las salidas A2, A1, A0 indican la línea activada de mayor prioridad. A2 es el MSB, A0 es el LSB. El valor de la salida debe validarse con el estado de GS (debe ser 0 para que la salida codificada se lea correctamente).

En la siguiente descripción las entradas están agrupadas en la señal i y el código de salida se obtiene en la señal a. En el vector i el bit de mayor prioridad es i(7) y el bit de menor prioridad es i(0).

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

ENTITY Codificador74148 IS
    PORT
    (   ei  : IN    STD_LOGIC;
        i   : IN    STD_LOGIC_VECTOR(0 TO 7);
        a   : OUT   STD_LOGIC_VECTOR(2 DOWNTO 0);
        eo,
        gs  : OUT   STD_LOGIC);
END Codificador74148;
```

En la asignación condicional de la salida **a**, se compara cada bit de entrada con cero (comenzando con el de mayor prioridad) y además se verifica que **ei** sea 0. No se evalúa que i(0) sea cero, pues el valor 1111, colocado como valor por defecto, corresponde con cero o con la desactivación de **ei**. Para saber que el valor correspondiente a 1111 es correcto, se debe revisar el valor de **gs**. Las funciones lógicas para **eo** y **gs** se obtienen de la tabla de verdad.

```
ARCHITECTURE condicional OF Codificador74148 IS
BEGIN

    a <= "000" WHEN i(7) = '0' AND ei = '0' ELSE
         "001" WHEN i(6) = '0' AND ei = '0' ELSE
         "010" WHEN i(5) = '0' AND ei = '0' ELSE
         "011" WHEN i(4) = '0' AND ei = '0' ELSE
         "100" WHEN i(3) = '0' AND ei = '0' ELSE
         "101" WHEN i(2) = '0' AND ei = '0' ELSE
         "110" WHEN i(1) = '0' AND ei = '0' ELSE
         "111";

    gs <=    '1' WHEN ei = '1' OR i = "11111111"
        ELSE '0';

    eo <=    '0' WHEN ei = '0' AND i = "11111111"
        ELSE '1';

END condicional;
```

Una simulación para este circuito se muestra en la figura 11.

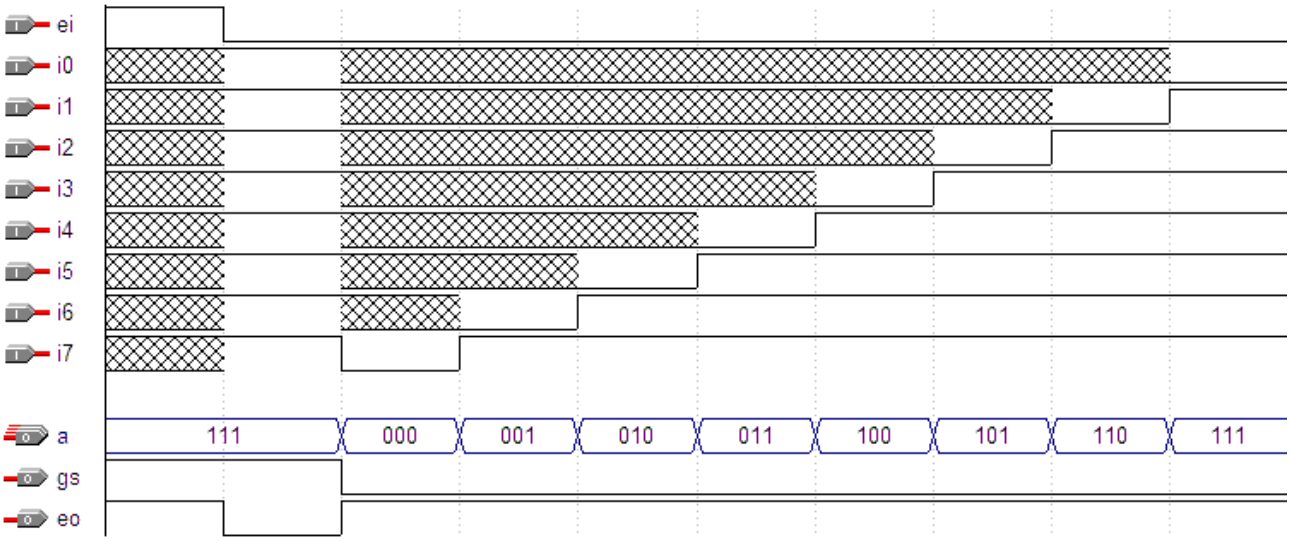


Figura 11. Un cronograma para la simulación VHDL del modelo del circuito integrado 74LS148.

Codificador de 16 a 4 líneas

Un codificador con prioridad de 16 a 4 líneas, con entradas y salidas activas en el nivel bajo, puede realizarse a partir de dos codificadores 74LS148 y cuatro puertas AND, tal como se muestra en la figura 12. Este circuito puede realizarse extendiendo la descripción previa del codificador 74LS148, mas en esta oportunidad se mostrará la descripción estructural del circuito utilizando dos codificadores 74LS148 y funciones AND.

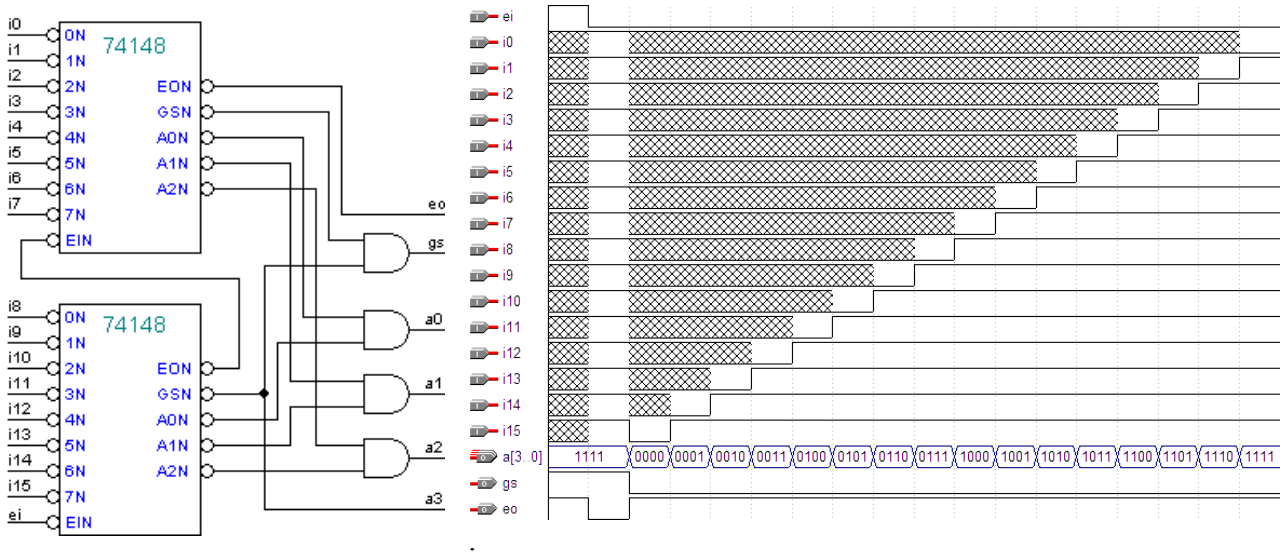


Figura 12. Diagrama esquemático y simulación de un codificador de 16 a 4 líneas.

La entidad para este circuito resulta:

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

ENTITY Codificador16a4 IS
    PORT
    (   ei  : IN    STD_LOGIC;
        i   : IN    STD_LOGIC_VECTOR(0 TO 15);
        a   : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0);
        eo,
        gs  : OUT   STD_LOGIC);
END Codificador16a4;
```

En la arquitectura se declara el componente Codificador74148. De este componente se crean dos instancias, un codificador para los bits de mayor prioridad (8 a 15) y otro para los de menor prioridad (0 a 7). La señal interna **eo1** es la salida de habilitación del codificador de mayor prioridad que se conecta a la entrada de habilitación del de menor prioridad. La señal **gs1** es la salida señalizadora de grupo del codificador de menor prioridad, y **gs2** es la salida del de mayor prioridad. Las señales **x** e **y** son las salidas de cada codificador.

```
ARCHITECTURE condicional OF Codificador16a4 IS

    COMPONENT Codificador74148 IS
    PORT
    (   ei  : IN    STD_LOGIC;
        i   : IN    STD_LOGIC_VECTOR(0 TO 7);
        a   : OUT   STD_LOGIC_VECTOR(2 DOWNTO 0);
        eo,
        gs  : OUT   STD_LOGIC);
    END COMPONENT;

    SIGNAL x, y: STD_LOGIC_VECTOR(2 DOWNTO 0);
    SIGNAL eo1, gs1, gs2: STD_LOGIC;
```

En el cuerpo de la arquitectura se crean dos instancias del componente y se completan las conexiones con operadores AND. Note que las operaciones AND pueden realizarse también con vectores de bits. En tal caso se realizan tantas operaciones AND como bits contengan los vectores, con las operaciones efectuándose bit a bit.

```
BEGIN

    UDe0a7: Codificador74148
    PORT MAP (eo1, i(0 TO 7), y, eo, gs1);

    UDe8a15: Codificador74148
    PORT MAP (ei, i(8 TO 15), x, eo1, gs2);

    a(3) <= gs2;
    a(2 DOWNTO 0) <= x AND y;
    gs <= gs1 and gs2;

END condicional;
```

Ejercicio 6

Realice la descripción VHDL del circuito anterior ampliando la descripción del codificador 74148, considerando una entrada de 16 bits en lugar de 8 bits.

Codificador de N a M líneas

Un codificador de N a M líneas puede realizarse fácilmente con la sentencia FOR – LOOP de VHDL, la cual permite realizar una descripción secuencial iterativa de otras sentencias del lenguaje. Esta sentencia solamente puede ir dentro de un proceso.

Su sintaxis básica es:

etiqueta:
FOR indice IN rango LOOP
sentencias;
END LOOP [etiqueta];

Para ilustrar su empleo, se creará un codificador de prioridad de 3 a 2 líneas. Para realizar la iteración se crea un arreglo de tres bits, donde el primer elemento es la entrada de menor prioridad, el segundo de prioridad intermedia y el tercero de mayor prioridad. Luego se explora iterativamente cada elemento con un índice. Si el elemento indizado está activado se obtiene su código como salida. Si más de un elemento está activado la última asignación será la que tenga efecto.

En esta descripción se utiliza la función CONV_STD_LOGIC_VECTOR para convertir un número entero a número binario. Esta función recibe como parámetros un número entero y la cantidad de bits para la conversión a binario. La entidad para este circuito es como antes, mas hay que incluir el paquete std_logic_arith, donde está definida la función CONV_STD_LOGIC_VECTOR.

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

ENTITY Codificador3a2 IS
    PORT
    (   a, b, c : IN    STD_LOGIC;
        y       : OUT   STD_LOGIC_VECTOR(1 DOWNTO 0)
    );
END Codificador3a2;
```

Agrupar los bits de entrada de menor a mayor índice en orden ascendente de prioridad.
 	
Asignar a la salida un valor por defecto
Desde el primer hasta el último elemento, repetir:
Si la entrada está activada
Asignar a la salida el índice equivalente en binario
	`ARCHITECTURE iterador OF Codificador3a2 IS`
	`  SIGNAL d: STD_LOGIC_VECTOR(1 TO 3);`
	`BEGIN`
	
	`  d <= c & b & a;`
	
	`  PROCESS(d)`
	`  BEGIN`
	`  y <= (others => '0');`
	`  FOR i IN 1 TO 3 LOOP`
	`  IF d(i) = '1' THEN`
	`  y <= CONV_STD_LOGIC_VECTOR(i, 2);`
	`  END IF;`
	`  END LOOP;`
	`  END PROCESS;`
	
	`END iterador;`

Para generalizar la descripción a un codificador de N a M líneas se crean los parámetros en la sección GENERIC y se les asigna valores por defecto. En este ejemplo se están asignando 8 bits de entrada (N) y 4 bits de salida (M). La salida de este codificador es un valor binario entre 1 y N, por lo que para codificar valores entre 1 y 8 son necesarios 4 bits.

En el cuerpo de la arquitectura el lazo se realiza desde 1 hasta N. En la conversión de entero a binario se indica convertir el índice i a M bits si el valor a(i) es 1. Como el procedimiento continua hasta el último bit del vector a, solamente la última asignación tendrá validez.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity CodificadorNaM is
    generic(
        N: integer:=8;      -- bits de entrada
        M: integer:= 4);    -- bits de salida
    port
    (   a   : in    std_logic_vector(1 to N);
        y   : out   std_logic_vector(M - 1 downto 0)
    );
end CodificadorNaM;

architecture a of CodificadorNaM is
begin

    process(a)
    begin
        y <= (others => '0');
        for i in 1 to N loop
            detector:
            if a(i) = '1' then
                y <= conv_std_logic_vector(i, M);
            end if;
        end loop;
    end process;

end a;
```

Ejercicio 7

Utilice el CodificadorNaM para crear un codificador de 10 a 4 líneas. Las entradas se ubican en las líneas indizadas de 1 a 10 y la salida es un número de 4 bits.

Conclusiones

1. Varios tipos de codificadores pueden describirse fácilmente en VHDL utilizando asignaciones condicionales, selectivas y procesos con IF-ELSIF-ELSE y CASE.

2. Una manera de crear codificadores de N a M líneas es con la sentencia FOR – LOOP dentro de un proceso.

3. En el siguiente capítulo se estudiarán varias descripciones de descodificadores.