USO DE UN DISPLAY MATRICIAL DE 8x8 CON VHDL

Juan Antonio Jaramillo Gómez, Dr., UPIITA-IPN

jantonioj@yahoo.com, jjaramillo@ipn.mx.

Mirna Salmerón Guzmán, M. en C., UPIITA-IPN

yupisg2@yahoo.com.mx.

Brahim El Filali, Dr., UPIITA-IPN

braelf@hotmail.com

Resumen

En la vida hay que comunicarse a través de la información y esta necesita distintos medios para llevarse a cabo. En este artículo se presenta un dispositivo electrónico que realiza esta tarea, en un display matricial de 8x8, con envío de información en los renglones y control de barrido en las columnas. El arreglo matricial del dispositivo comercial tiene los ánodos en los renglones y los cátodos en las columnas. Se muestran 3 ejemplos de uso (imágenes fijas y en movimiento de símbolos y números), implementados en una tarjeta de desarrollo Nexys 2 con un FPGA y lenguaje descriptivo de hardware VHDL utilizando el programa ISE de Xilinx. Los códigos completos y los videos se muestran en ligas el final del documento.

Abstract

In life you have to communicate through information and it needs different means to be carried out. This article presents an electronic device that performs this task, using an 8x8 matrix display, with sending information in rows and control of sweeping in the columns. The matrix arrangement of the commercial device has the anodes in the rows and the cathodes in the columns. It shows 3 examples of use (fix and moving images of symbols and numbers), implemented in a Nexys 2 development board with an FPGA and VHDL hardware description language using the Xilinx ISE program. Full codes and videos are displayed in leagues at the end of the document.

Introducción

La visualización de información en displays es muy común para dar mensaje, avisos y anuncios, tanto para el entretenimiento, señalizaciones, indicaciones o el funcionamiento de algo. Existen distintos tipos de elementos de visualización (display) como los mostrados a continuación y los presentados en la figura 1, además de estar disponibles en distintos tamaños, formas, colores y tecnologías.

- Display matricial
- Display de cristal líquido (LCD, GLCD)
- Display de segmentos (7-seg, 8-seg, 15-seg, 17-seg, etc.)
- Display programable (PD)
- Pantallas
- Proyectores

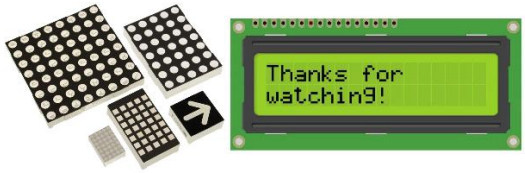

Figura 1. Distintos tipos de displays.

Aquí se hará el uso de un display matricial de 8x8, por lo que la forma de utilizarlo comienza con saber qué tipo de display se tiene, uno de ánodo común o uno de cátodo común. Lo siguiente que hay que saber es su distribución de terminales, esta información se obtiene de las hojas de especificaciones del componente que se tenga, como los mostrados en la figura 2, que muestra la distribución de pines para cada tipo de configuración.

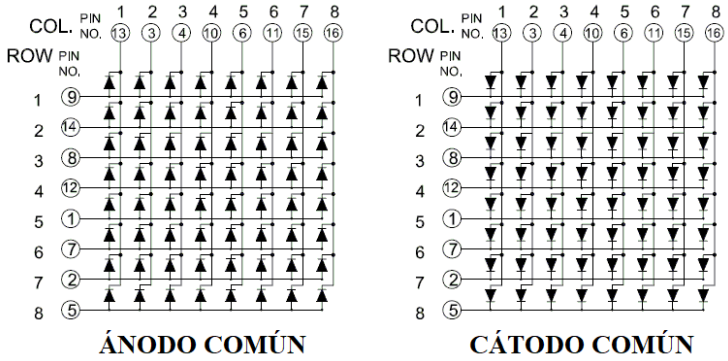

Figura 2. Diagrama interno de un display matricial de 8x8.

Lo último que hay que saber es si el display se conecta de forma directa o si el display tiene un controlador. En el primer punto se puede conectar en proto o en tarjeta de prototipado rápido o una placa de PCB. En el segundo hay que conocer el tipo de dispositivo controlador y en su caso, el protocolo de comunicación. Uno de los dispositivos más comunes es el MAX7219, que es un controlador serial y se puede conectar en cascada para formar una matriz más grande, mostrados en la figura 3.

Figura 3. Tipos de arreglos y controlador de display matricial en PCB.

El presente artículo muestra la implementación de imágenes fijas y en movimiento en un display matricial de 8x8 conectado directamente a la tarjeta de desarrollo Nexys 2, con un display hecho en prototipado rápido con *wire solder* en placa perforada. Los códigos o partes de estos pueden reutilizarse para implementar otras figuras en otras aplicaciones comerciales o industriales.

Desarrollo

Entrando directamente en materia, se presentan los resultados en fotos y videos con ligas para la demostración del funcionamiento de tres códigos, también en ligas, que se implementaron en la tarjeta Nexys 2 y un display matricial de 8x8 como módulo periférico o módulo externo (ver figura 4), hecho en placa perforada y la técnica de "*wire solder*" que se acopla directamente a los puertos de la tarjeta. Las aplicaciones desarrolladas son:

- 4 imágenes (flecha, rombo, signo de + y carita feliz)
- Contador ascendente del 0 al 10
- Imagen fija y flecha en movimiento izquierda o derecha con sonido

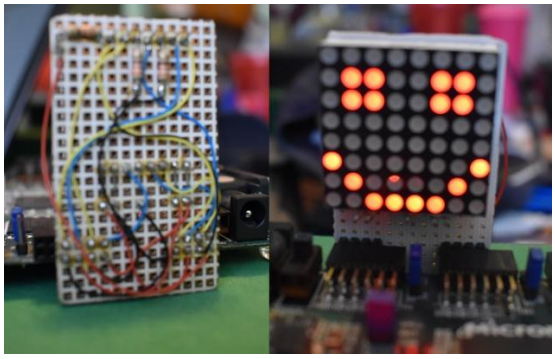

Figura 4. Módulo periférico de display matricial de 8x8 en "wire solder".

Un display matricial puede mostrar una o varias imágenes fijas o imágenes en movimiento. Las imágenes pueden ser:

- Símbolos
- Números
- Letras

Una vez sabiendo lo que se desea mostrar se pasa al diseño de la(s) imagen(es), como se muestra en la figura 5, hay que elegir entre que se desean mostrar, sin olvidar que uno mismo es el diseñador.

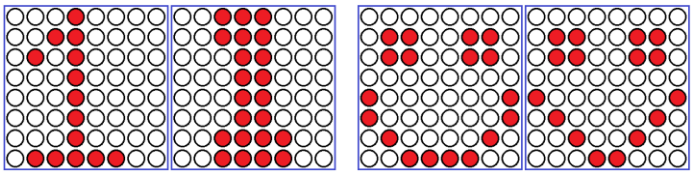

Figura 5. Se presentan dos diseños del número 1 y dos diseños de carita feliz.

A continuación, se presentan los resultados de los diseños implementados, resaltando que los códigos y videos se muestran en ligas al final del artículo.

El primer código implementado es una secuencia de símbolos mostrados un tiempo menos de un segundo y se ciclan. Los símbolos son rombo, sígno de más “+”, flecha hacia arriba y una carita feliz, como se muestra en la figura 6.

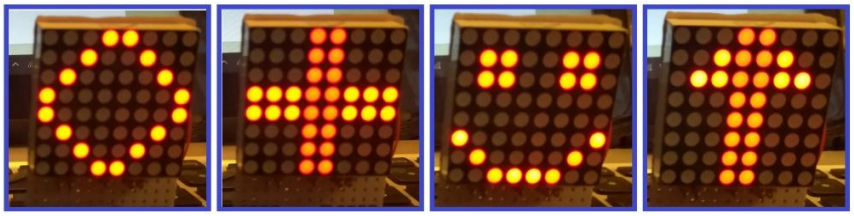

Figura 6. Se presentan los símbolos del rombo, suma, carita feliz y flecha hacia arriba.

El segundo código implementado es el corrimiento de los números del 0 al 10, como se puede observar en la figura 7.

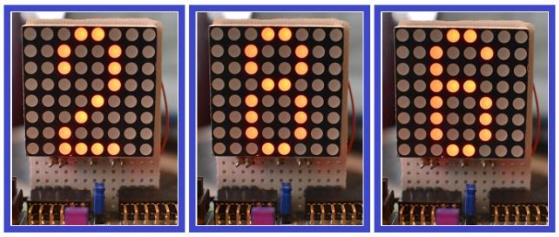

Figura 7. Se presentan tres de los once números implementados.

En el tercer código se implementaron flechas en movimiento que corren de izquierda a derecha y de derecha hacia la izquierda, con dos distintos sonidos (beep) que cambian con respecto al sentido de la flecha. También cuenta con botones para quitar las flechas y poner un cuadrado con puntos en las esquinas, un botón para cambiar de sentido las flechas y un botón para apagar el sonido. Todo esto se aprecia directamente en los *videos* (símbolos: Disp8x8simb.mp4, números del 0 al 10: Disp8x8num.mp4, flechas: Disp8x8flechas.mp4). En la figura 8 se muestran las fotos de las secuencias de las flechas.

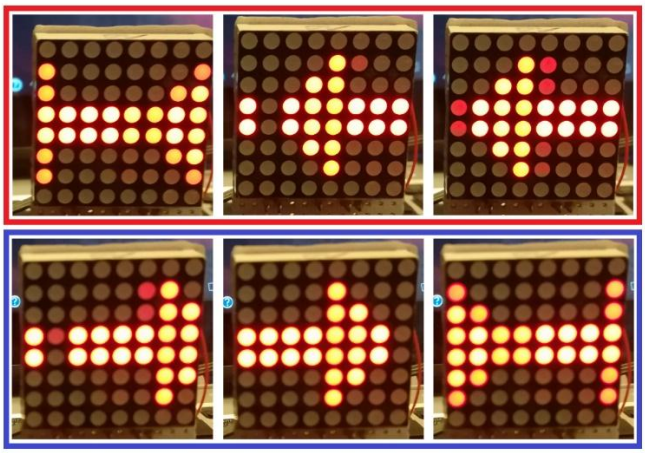

Figura 8. Se presentan las imágenes del funcionamiento de las flechas,

(arriba va a la izquierda, abajo va a la derecha).

Por último, como ya se ha mencionado, al final del documento se encuentran los códigos implementados en ligas para su revisión, consulta o reciclaje de los códigos en listado de VHDL y el archivo de restricciones de usuario (UCF *user constrain file*).

Conclusiones

Se ha presentado la implementación de tres distintos códigos para el uso de un display matricial de 8x8 en un prototipado rápido, con códigos en VHDL y el UCF. Las fotos y los videos son la evidencia demostrativa de su funcionamiento. Los tipos de aplicación contemplan tanto imágenes fijas como en movimiento, mostrando símbolos, letras números y animaciones, y las aplicaciones dependen del usuario.

Referencias y Recursos electrónicos

Haskell, Richard E. y Hanna, Darrin M., Digital Design using FPGA Boards VHDL, LBE Books, pp. 93-108, 203-209.

Haskell, Richard E. y Hanna, Darrin M., Digital Design using FPGA Boards Verilog, LBE Books, pp. 109-119, 185-190.

Reference manual Nexys 2, recuperado en octubre 2017, disponible en: http://www.digilentinc.com

Teclado matricial y matriz de leds, recuperado en octubre 2017, disponible en: http://www.embebidos-cidetec.com.mx/profesores/jcrls/doctos/, http://embebidos-cidetec.com.mx/profesores/jcrls/doctos/mtc_p8_2013.pdf

VHDL code for led matrix, recuperado en octubre 2017, disponible en: https://forums.xilinx.com/t5/Spartan-Family-FPGAs/VHDL-code-for-a-LED-matrix/td-p/235868

Códigos para anexarse en ligas:

primer código**********

```
-- Este código presenta 4 figuras cicladas en el display matricial de 8x8.
-- Los renglones van a resistencia y de ahí a los ánodos.
--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity matrix4Fig is
   Generic (N: integer:=15; M: integer:=26); -- N valor de bits del divisor
   Port ( clk : in  STD_LOGIC; -- reloj de 50MHz
          R,C : out STD_LOGIC_VECTOR (8 downto 1)); -- Renglones y Columnas
end matrix4Fig;

architecture matrix4Fig of matrix4Fig is

--señales

signal clkdiv: std_logic_vector (M downto 0); --divisor de M+1 bits
signal barrido: std_logic_vector (2 downto 0); --contador de tres bits que sirve para el
                                                barrido del arreglo
type arreglo is array (1 to 8) of std_logic_vector(7 downto 0);
signal tabla: arreglo; --señal que recibe las cuatro figuras (tabla1,2,3,4) para ciclarse

--constantes

constant tabla1 : arreglo :=(  -- datos de la flecha arriba
                    "00000000",
                    "00000100",
                    "00000110",
                    "11111111",
```

```
                "11111111",
                "00000110",
                "00000100",
                "00000000");

    constant tabla2 : arreglo :=(  -- datos del rombo
                "00011000",
                "00100100",
                "01000010",
                "10000001",
                "10000001",
                "01000010",
                "00100100",
                "00011000");

    constant tabla3 : arreglo :=(  -- datos del símbolo suma "+"
                "00011000",
                "00011000",
                "00011000",
                "11111111",
                "11111111",
                "00011000",
                "00011000",
                "00011000");
```

```
constant tabla4 : arreglo :=(  -- datos de la carita
                    "00100000",
                    "01000110",
                    "10000110",
                    "10000000",
                    "10000000",
                    "10000110",
                    "01000110",
                    "00100000");

signal tempo: std_logic_vector(1 downto 0);
begin

-- proceso del divisor cldiv

divisor: process (clk)
begin
        if clk'event and clk='1' then
        clkdiv <= clkdiv + 1;--contador de M bits
        end if;
end process divisor;

--manda los datos del display
asigna: process (clkdiv(M), barrido)--, clkdiv(M))
```

```
begin
tempo <= clkdiv(M downto M-1);
--          esta asignación funciona igual que clkdiv <= clkdiv + 1
            barrido <= clkdiv(N downto N-2);

-- cambio de las figuras para la señal tabla
if    tempo = "00" then tabla <= tabla1;
elsif tempo = "01" then tabla <= tabla2;
elsif tempo = "10" then tabla <= tabla3;
elsif tempo = "11" then tabla <= tabla4;
else                    tabla <= tabla4;
end if;

--se mandan los datos a los renglones y las columnas con el contador barrido
                    case barrido is
                    when o"0" =>  R <= tabla(1); C <= "01111111";
                    when o"1" =>  R <= tabla(2); C <= "10111111";
                    when o"2" =>  R <= tabla(3); C <= "11011111";
                    when o"3" =>  R <= tabla(4); C <= "11101111";
                    when o"4" =>  R <= tabla(5); C <= "11110111";
                    when o"5" =>  R <= tabla(6); C <= "11111011";
                    when o"6" =>  R <= tabla(7); C <= "11111101";
                    when o"7" =>  R <= tabla(8); C <= "11111110";
                    when others =>R <= tabla(1); C <= "00000000";
```

```
                end case;


end process asigna;


end matrix4Fig;
```

Archivo de restricciones de usuario del primer código.

```
#asignación de pines para el display de 8x8 de la nexys2


#reloj

net "CLK" loc = "B8" ;


#renglones

net "C(1)" loc = "L15" ;        # a JA1

net "C(2)" loc = "K12" ;        # a JA2

net "C(3)" loc = "L17" ;        # a JA3

net "C(4)" loc = "M15" ;        # a JA4

net "C(5)" loc = "M13" ;        # a JB1

net "C(6)" loc = "R18" ;        # a JB2

net "C(7)" loc = "R15" ;        # a JB3

net "C(8)" loc = "T17" ;        # a JB4


#columnas

net "R(1)" loc = "K13" ;        # a JA7
```

```
net "R(2)" loc = "L16" ;          # a JA8
net "R(3)" loc = "M14" ;          # a JA9
net "R(4)" loc = "M16" ;          # a JA10
net "R(5)" loc = "P17" ;          # a JB7
net "R(6)" loc = "R16" ;          # a JB8
net "R(7)" loc = "T18" ;          # a JB9
net "R(8)" loc = "U18" ;          # a JB10
```

Códigos para anexarse en ligas:

segundo código**********

```
-- Este código presenta los números del 0 al 10 en un display de 8x8
-- Los renglones van a resistencia y de ahi a los ánodos.
--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity matrixNum is
    Generic (N: integer:=15; M: integer:=25); -- N valor de bits del divisor
   Port ( clk : in  STD_LOGIC; -- reloj de 50MHz
          R,C : out STD_LOGIC_VECTOR (8 downto 1)); -- Renglones y Columnas
end matrixNum;
```

```
architecture matrixNum of matrixNum is

--señales

signal clkdiv: std_logic_vector (M downto 0); --divisor de M+1 bits
signal barrido: std_logic_vector (2 downto 0); --contador de 3 bits que sirve para el barrido
type arreglo is array (1 to 8) of std_logic_vector(7 downto 0);
signal tabla : arreglo;

--constantes

constant tabla0 : arreglo :=(  -- datos del numero 0
        "00000000",  --"00011000"
        "00000000",  --"00100100"
        "01111110",  --"00100100"
        "10000001",  --"00100100"
        "10000001",  --"00100100"
        "01111110",  --"00100100"
        "00000000",  --"00100100"
        "00000000"); --"00011000");
constant tabla1 : arreglo :=(  -- datos del numero 1
        "00000000",  --"00001000"
        "00000000",  --"00011000"
        "00000000",  --"00001000"
        "10000010",  --"00001000"
```

```
        "11111111",  --"00001000"
        "10000000",  --"00001000"
        "00000000",  --"00001000"
        "00000000"); --"00011100");
    constant tabla2 : arreglo :=(  -- datos del numero 2
        "00000000",  --"00011000"
        "00000000",  --"00100100"
        "11000110",  --"00100100"
        "10100001",  --"00000100"
        "10010001",  --"00001000"
        "10001110",  --"00010000"
        "00000000",  --"00100000"
        "00000000"); --"00111100");
    constant tabla3 : arreglo :=(  -- datos del numero 3
        "00000000",  --"00011000"
        "00000000",  --"00100100"
        "01000010",  --"00000100"
        "10000001",  --"00001000"
        "10001001",  --"00000100"
        "01110110",  --"00000100"
        "00000000",  --"00100100"
        "00000000"); --"00011000");
    constant tabla4 : arreglo :=(  -- datos del numero 4
        "00000000",  --"00100100"
        "00000000",  --"00100100"
```

```
        "00001111",  --"00100100"
        "00001000",  --"00111100"
        "00001000",  --"00000100"
        "11111111",  --"00000100"
        "00000000",  --"00000100"
        "00000000"); --"00000100");
    constant tabla5 : arreglo :=(  -- datos del numero 5
        "00000000",  --"00111100"
        "00000000",  --"00100000"
        "01000111",  --"00111000"
        "10000101",  --"00000100"
        "10000101",  --"00000100"
        "01111001",  --"00000100"
        "00000000",  --"00100100"
        "00000000"); --"00011000");
    constant tabla6 : arreglo :=(  -- datos del numero 6
        "00000000",  --"00011000"
        "00000000",  --"00100100"
        "01111110",  --"00100000"
        "10001001",  --"00111000"
        "10001001",  --"00100100"
        "01110010",  --"00100100"
        "00000000",  --"00100100"
        "00000000"); --"00011000");
    constant tabla7 : arreglo :=(  -- datos del numero 7
```

```
        "00000000",  --"00111100"
        "00000000",  --"00000100"
        "10000001",  --"00000100"
        "01100001",  --"00001000"
        "00011001",  --"00001000"
        "00000111",  --"00010000"
        "00000000",  --"00010000"
        "00000000"); --"00100000");
constant tabla8 : arreglo :=(  -- datos del numero 8
        "00000000",  --"00011000"
        "00000000",  --"00100100"
        "01110110",  --"00100100"
        "10001001",  --"00011000"
        "10001001",  --"00100100"
        "01110110",  --"00100100"
        "00000000",  --"00100100"
        "00000000"); --"00011000");
constant tabla9 : arreglo :=(  -- datos del numero 9
        "00000000",  --"00011000"
        "00000000",  --"00100100"
        "01000110",  --"00100100"
        "10001001",  --"00011100"
        "10001001",  --"00000100"
        "01111110",  --"00000100"
        "00000000",  --"00100100"
```

```
        "00000000"); --"00011000");
constant tabla10 : arreglo :=(  -- datos del numero 10
        "00000010",  --"01001100"
        "11111111",  --"11010010"
        "00000000",  --"01010010"
        "01111110",  --"01010010"
        "10000001",  --"01010010"
        "10000001",  --"01010010"
        "01111110",  --"01010010"
        "00000000"); --"01001100");
signal conta: integer range 0 to 10; --señal para contar del 0 al 10

begin

-- proceso del divisor cldiv

divisor: process (clk)
begin
        if clk'event and clk='1' then
        clkdiv <= clkdiv + 1;
        end if;
end process divisor;

--manda los datos del display
asigna: process (clkdiv(M),clkdiv(N),barrido, conta) --, clkdiv(M))
```

```
begin

--          esta asignación funciona igual que clkdiv <= clkdiv + 1
            barrido <= clkdiv(N downto N-2);

--conteo de 0 a 10
 if rising_edge(clkdiv(M)) then
   if conta=10 then conta <= 0;
            else conta <= conta + 1;
            end if;
 end if;

--selección de la tabla a partir del valor de conta
     case conta is
                    when 0  => tabla <= tabla0;
                    when 1  => tabla <= tabla1;
                    when 2  => tabla <= tabla2;
                    when 3  => tabla <= tabla3;
                    when 4  => tabla <= tabla4;
                    when 5  => tabla <= tabla5;
                    when 6  => tabla <= tabla6;
                    when 7  => tabla <= tabla7;
                    when 8  => tabla <= tabla8;
                    when 9  => tabla <= tabla9;
                    when 10  => tabla <= tabla10;
```

```
                end case;

--se mandan los datos a los renglones y las columnas con el contador barrido
                case barrido is
                when o"0" =>  R <= tabla(8); C <= "01111111";
                when o"1" =>  R <= tabla(7); C <= "10111111";
                when o"2" =>  R <= tabla(6); C <= "11011111";
                when o"3" =>  R <= tabla(5); C <= "11101111";
                when o"4" =>  R <= tabla(4); C <= "11110111";
                when o"5" =>  R <= tabla(3); C <= "11111011";
                when o"6" =>  R <= tabla(2); C <= "11111101";
                when o"7" =>  R <= tabla(1); C <= "11111110";
                when others =>R <= tabla(1); C <= "00000000";
                end case;

end process asigna;

end matrixNum;
```

Archivo de restricciones de usuario del segundo código.

```
# asignación de pines para el display de 8x8 de la nexys2 del
# reloj y en los conectores de expansión JA y JB

#reloj
```

```
net "CLK" loc = "B8" ;


#renglones
net "C(1)" loc = "L15" ;        # a JA1
net "C(2)" loc = "K12" ;        # a JA2
net "C(3)" loc = "L17" ;        # a JA3
net "C(4)" loc = "M15" ;        # a JA4
net "C(5)" loc = "M13" ;        # a JB1
net "C(6)" loc = "R18" ;        # a JB2
net "C(7)" loc = "R15" ;        # a JB3
net "C(8)" loc = "T17" ;        # a JB4

#columnas
net "R(1)" loc = "K13" ;        # a JA7
net "R(2)" loc = "L16" ;        # a JA8
net "R(3)" loc = "M14" ;        # a JA9
net "R(4)" loc = "M16" ;        # a JA10
net "R(5)" loc = "P17" ;        # a JB7
net "R(6)" loc = "R16" ;        # a JB8
net "R(7)" loc = "T18" ;        # a JB9
net "R(8)" loc = "U18" ;        # a JB10
```

Códigos para anexarse en ligas:

tercer código**********

```
-- Este código presenta una flecha en un display matricial
-- de 8x8.
-- Los renglones van a resistencia y de ahí a los ánodos.
--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity matrixArrowMov is
        Generic (N: integer:=15; M: integer:=24); -- M es para el divisor y N para el barrido
  Port ( clk,dir,hold,sonidoSW : in  STD_LOGIC; -- reloj de 50MHz, dirección, hold y sonido
         sonidoOUT: out std_logic; --salida de sonido para una bocina de alta impedancia
         sonidoOUT2: out std_logic; --salida de sonido para una bocina de alta impedancia
         R,C : out STD_LOGIC_VECTOR (8 downto 1)); -- Renglones y Columnas
end matrixArrowMov;

architecture matrixArrowMov of matrixArrowMov is

--señales
signal clkdiv: std_logic_vector (M downto 0); --divisor de M+1 bits
signal barrido: std_logic_vector (2 downto 0); --contador de 3 bits para el barrido del arreglo
type arreglo is array (1 to 8) of std_logic_vector(7 downto 0); --declaración de la matriz 8x8
signal tabla: arreglo; --señal que recibe las cuatro figuras (tabla1,2,3,4) para ciclarse
```

```
--constantes
constant tabla1 : arreglo :=(  -- datos de la flecha arriba
        "00011000",  --"00000000",
        "00111100",  --"00000100",
        "01111110",  --"00000110",
        "00011000",  --"01111111",
        "00011000",  --"01111111",
        "00011000",  --"00000110",
        "00011000",  --"00000100",
        "00000000"); --"00000000");

constant tabla2 : arreglo :=(  -- datos de la flecha arriba
        "00111100",  --"00000000",
        "01111110",  --"00000010",
        "00011000",  --"00000011",
        "00011000",  --"10111111",
        "00011000",  --"10111111",
        "00011000",  --"00000011",
        "00000000",  --"00000010",
        "00011000"); --"00000000");
constant tabla3 : arreglo :=(  -- datos de la flecha arriba
        "01111110",  --"00000000",
        "00011000",  --"00000001",
        "00011000",  --"10000001",
```

```
        "00011000",  --"11011111",
        "00011000",  --"11011111",
        "00000000",  --"10000001",
        "00011000",  --"00000001",
        "00111100"); --"00000000");
constant tabla4 : arreglo :=(  -- datos de la flecha arriba
        "00011000",  --"00000000",
        "00011000",  --"10000000",
        "00011000",  --"11000000",
        "00011000",  --"11101111",
        "00000000",  --"11101111",
        "00011000",  --"11000000",
        "00111100",  --"10000000",
        "01111110"); --"00000000");
constant tabla5 : arreglo :=(  -- datos de la flecha arriba
        "00011000",  --"00000000",
        "00011000",  --"01000000",
        "00011000",  --"01100000",
        "00000000",  --"11110111",
        "00011000",  --"11110111",
        "00111100",  --"01100000",
        "01111110",  --"01000000",
        "00011000"); --"00000000");
constant tabla6 : arreglo :=(  -- datos de la flecha arriba
        "00011000",  --"00000000",
```

```
        "00011000",  --"00100000",
        "00000000",  --"00110000",
        "00011000",  --"11111011",
        "00111100",  --"11111011",
        "01111110",  --"00110000",
        "00011000",  --"00100000",
        "00011000"); --"00000000");
    constant tabla7 : arreglo :=(  -- datos de la flecha arriba
        "00011000",  --"00000000",
        "00000000",  --"00010000",
        "00011000",  --"00011000",
        "00111100",  --"11111101",
        "01111110",  --"11111101",
        "00011000",  --"00011000",
        "00011000",  --"00010000",
        "00011000"); --"00000000");
    constant tabla8 : arreglo :=(  -- datos de la flecha arriba
        "00000000",  --"00000000",
        "00011000",  --"00001000",
        "00111100",  --"00001100",
        "01111110",  --"11111110",
        "00011000",  --"11111110",
        "00011000",  --"00001100",
        "00011000",  --"00001000",
        "00011000"); --"00000000");
```

```
constant tabla9 : arreglo :=(  -- datos de puntos en esquinas y cuadro
        "10000001",  --"00000000",
        "00000000",  --"00001000",
        "00111100",  --"00001100",
        "00100100",  --"11111110",
        "00100100",  --"11111110",
        "00111100",  --"00001100",
        "00000000",  --"00001000",
        "10000001"); --"00000000");

-- la señal tempo es un contador de 3 bits que asigna de forma temporal el valor
-- de las 8 tablas
signal tempo: std_logic_vector(2 downto 0);--
signal duracion: std_logic_vector(1 downto 0); --contador para el sonido

begin --comienza la arquitectura

-- proceso del divisor cldiv

divisor: process (clk)
begin
        if clk'event and clk='1' then
        clkdiv <= clkdiv + 1; --contador de M bits
        end if;
end process divisor;
```

```
--manda los datos del display
asigna: process (clkdiv(M), barrido, dir, hold)
begin
tempo <= clkdiv(M downto M-2);

--        esta asignación funciona igual que clkdiv <= clkdiv + 1
          barrido <= clkdiv(N downto N-2);

-- cambio de las figuras para la señal tabla
if     tempo = o"0" then tabla <= tabla1;
elsif tempo = o"1" then tabla <= tabla2;
elsif tempo = o"2" then tabla <= tabla3;
elsif tempo = o"3" then tabla <= tabla4;
elsif tempo = o"4" then tabla <= tabla5;
elsif tempo = o"5" then tabla <= tabla6;
elsif tempo = o"6" then tabla <= tabla7;
else                           tabla <= tabla8;
end if;

--se mandan los datos a los renglones y las columnas con el contador barrido
if hold='1' then          --manda cuadro con puntos en las esquinas
          case barrido is
                    when o"0" =>  R <= tabla9(1); C <= "11111110";
                    when o"1" =>  R <= tabla9(2); C <= "11111101";
```

```
                when o"2" =>  R <= tabla9(3); C <= "11111011";
                when o"3" =>  R <= tabla9(4); C <= "11110111";
                when o"4" =>  R <= tabla9(5); C <= "11101111";
                when o"5" =>  R <= tabla9(6); C <= "11011111";
                when o"6" =>  R <= tabla9(7); C <= "10111111";
                when o"7" =>  R <= tabla9(8); C <= "01111111";
                when others =>R <= tabla9(1); C <= "00000000";
        end case;

elsif dir='1' then
        case barrido is
                when o"0" =>  R <= tabla(1); C <= "11111110";
                when o"1" =>  R <= tabla(2); C <= "11111101";
                when o"2" =>  R <= tabla(3); C <= "11111011";
                when o"3" =>  R <= tabla(4); C <= "11110111";
                when o"4" =>  R <= tabla(5); C <= "11101111";
                when o"5" =>  R <= tabla(6); C <= "11011111";
                when o"6" =>  R <= tabla(7); C <= "10111111";
                when o"7" =>  R <= tabla(8); C <= "01111111";
                when others =>R <= tabla(1); C <= "00000000";
        end case;

else
        case barrido is
                when o"0" =>  R <= tabla(8); C <= "11111110";        --"11111110";
```

```
                when o"1" =>  R <= tabla(7); C <= "11111101";       --"11111101";
                when o"2" =>  R <= tabla(6); C <= "11111011";       --"11111011";
                when o"3" =>  R <= tabla(5); C <= "11110111";       --"11110111";
                when o"4" =>  R <= tabla(4); C <= "11101111";       --"11101111";
                when o"5" =>  R <= tabla(3); C <= "11011111";       --"11011111";
                when o"6" =>  R <= tabla(2); C <= "10111111";       --"10111111";
                when o"7" =>  R <= tabla(1); C <= "01111111";       --"01111111";
                when others =>R <= tabla(1); C <= "00000000";       --"00000000";
        end case;
end if;

end process asigna;

sonido: process(clkdiv(M), sonidoSW, dir, hold)
begin
duracion <= clkdiv(M downto M-1);

if sonidoSW = '0' or hold = '1' then sonidoOUT <= '0';        sonidoOUT2 <= '0'; --sin sonido
elsif dir = '1' then
        case duracion is
          when "00" => sonidoOUT <= clkdiv(15);   sonidoOUT2 <= clkdiv(15);
          when others => sonidoOUT <= '0'; sonidoOUT2 <= '0';
        end case;
else -- dir = '0' then
        case duracion is
```

```
            when "00" => sonidoOUT <= clkdiv(16);   sonidoOUT2 <= clkdiv(16);
            when "01" => sonidoOUT <= '0';          sonidoOUT2 <= '0'; --clkdiv(15);
            when "10" => sonidoOUT <= clkdiv(16);   sonidoOUT2 <= clkdiv(16);
            when "11" => sonidoOUT <= '0';          sonidoOUT2 <= '0'; --clkdiv(13);
            when others => sonidoOUT <= '0';        sonidoOUT2 <= '0';
          end case;
end if;
end process sonido;


end matrixArrowMov;
```

Archivo de restricciones de usuario del tercer código.

```
#asignación de pines para el display de 8x8 de la nexys2


#reloj, sentido y hold
net "CLK" loc = "B8" ;                  #al reloj de la nexys2

net "dir" loc = "G18" ;                 #a sw0, dir=1 sube, dir=0 baja
net "sonidoSW" loc = "K18" ;            #a sw2, activado en alto
net "hold" loc = "R17" ;                #a sw7, activado en alto

net "sonidoOUT" loc = "P18" ;           #a JD4, salida de sonido
net "sonidoOUT2" loc = "J14" ;          #a JD10, salida de sonido2, compartido con LD3
```

```
#renglones
net "C(1)" loc = "L15" ;        # a JA1JA
net "C(2)" loc = "K12" ;        # a JA2
net "C(3)" loc = "L17" ;        # a JA3
net "C(4)" loc = "M15" ;        # a JA4
net "C(5)" loc = "M13" ;        # a JB1 JB
net "C(6)" loc = "R18" ;        # a JB2
net "C(7)" loc = "R15" ;        # a JB3
net "C(8)" loc = "T17" ;        # a JB4

#columnas
net "R(1)" loc = "K13" ;        # a JA7JA
net "R(2)" loc = "L16" ;        # a JA8
net "R(3)" loc = "M14" ;        # a JA9
net "R(4)" loc = "M16" ;        # a JA10
net "R(5)" loc = "P17" ;        # a JB7 JB
net "R(6)" loc = "R16" ;        # a JB8
net "R(7)" loc = "T18" ;        # a JB9
net "R(8)" loc = "U18" ;        # a JB10
```