SISTEMAS ELECTRÓNICOS DIGITALES

TRABAJO VHDL

MAQUINA EXPENDEDORA

POLITÉCNICA

ALUMNOS:

Sara Rodríguez Fernández 54832

Víctor Rodríguez Sánchez 54838

Oscar Rudek 54844

FECHA: 19 de enero de 2022

PROFESOR: Luis Castedo

INDICE

1. INTRODUCCIÓN

El objetivo de este trabajo ha sido diseñar una máquina expendedora de refrescos. La máquina admite cuatro tipos de monedas: de 10 céntimos, 20 céntimos, 50 céntimos y 1 euro. Además, dispone de cuatro productos a elegir, todos de coste 1 euro.

Sin embargo, sólo admite el importe exacto, de forma que si introducimos una cantidad superior da un error y devolverá todo el dinero introducido. Por otro lado, cuando se llega al importe exacto del refresco (1 euro) se activará una señal para dar el producto.

Como entradas se emplean los botones como señales indicadoras de la moneda y señales indicadoras de producto, que se manejan con los interruptores. Por último, como salidas se tienen los mensajes que aparecen en el display, que son los de error y dinero introducido correcto, la cantidad de dinero introducida y el producto elegido. Además de la señal que simula el motor dispensador del producto.

2. DESCRIPCION DE LA ESTRATEGIA Y DESARROLLO DE ALGORITMOS PARA LA REALIZACCIÓN DEL TRABAJO

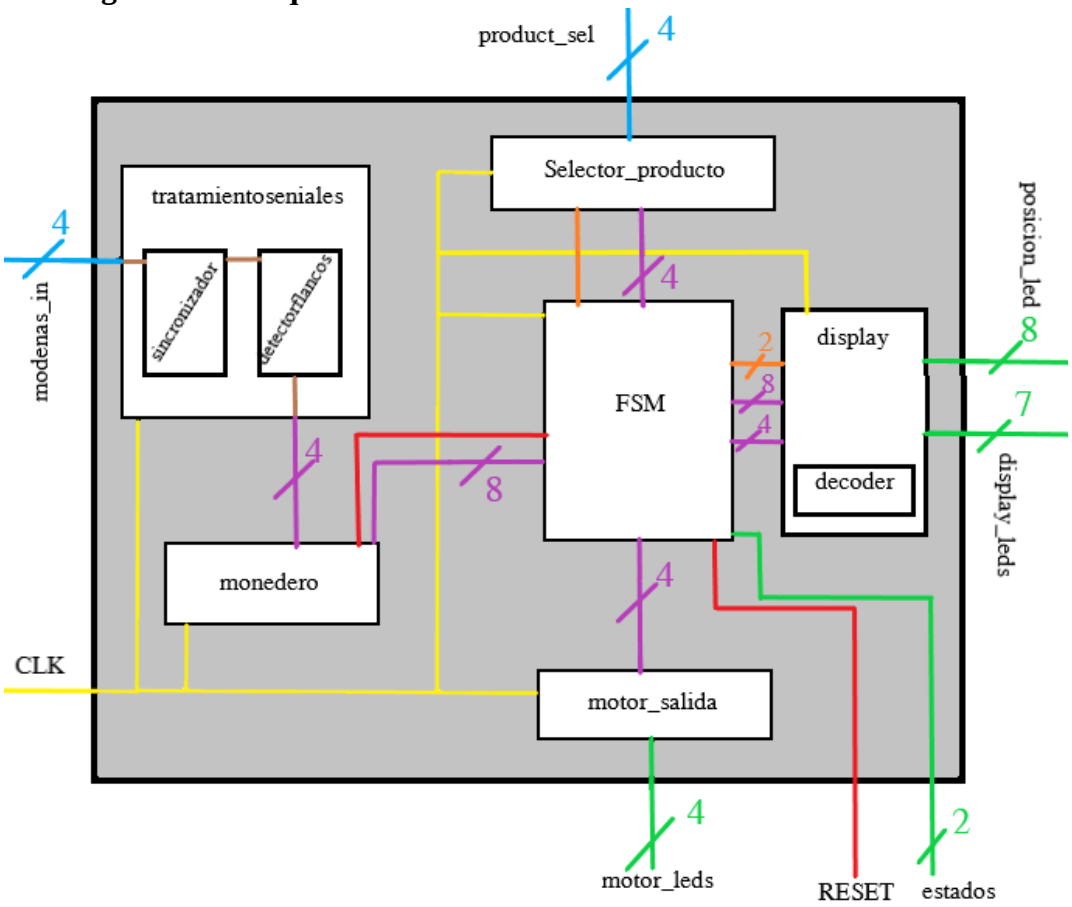
El sistema que hemos implementado se trata de un sistema síncrono, cuyo comportamiento está controlado por una máquina de estados finitos síncrona.

Hemos diseñado cuatro estados para determinar el funcionamiento de la máquina expendedora. En primer lugar, se parte del estado de reposo, de tal modo que, si se selecciona uno de los cuatros productos mediante los interruptores de la placa, tras un tiempo de transición entre estados, se pasa al estado uno y se envía el número del producto elegido al display. A continuación, en el estado uno se lee el valor del monedero, que aumenta en función del botón pulsado, que se corresponde con cada tipo de moneda, y se va mostrando en el display. Si se inserta la cantidad exacta, 100 céntimos, es decir 1 euro, pasará al estado dos, donde se activa el motor correspondiente al producto elegido para soltarlo encendiéndose su led, y aparecerá el mensaje de “FULL” en el display. De lo contrario, si se inserta una cantidad mayor pasará al estado tres de error, se devolverá el dinero, reseteando el monedero y aparecerá el mensaje de “ERROR”. Una vez pasado el tiempo de transición entre estados, desde el dos o el tres vuelve al estado cero de reposo.

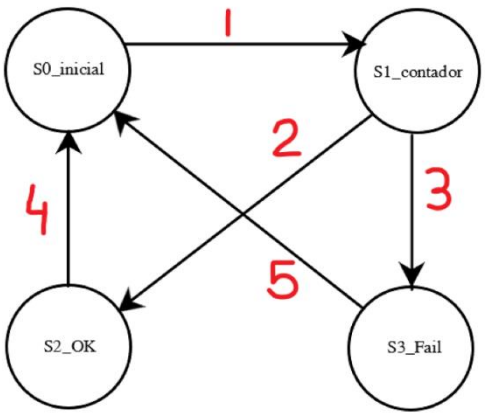
Por otra parte, a la hora de elaborar nuestro sistema hemos empleado la estrategia Top-Down, puesto que permite programar de forma sencilla e ir simulando poco a poco los distintos módulos del sistema. De este modo, lo hemos dividido en diez módulos sencillos: la máquina de estados (FSM), el selector del producto, el decodificador, el de tratamiento de señales de entrada, que incluye como componentes a el detector de flancos y el sincronizador, el bloque de la visualización (displays), el monedero, el del motor de la máquina, y el top que agrupa todos los bloques anteriores en un sistema.

3. DIAGRAMAS

3.1 Diagrama de bloques



3.2 Diagrama de estados



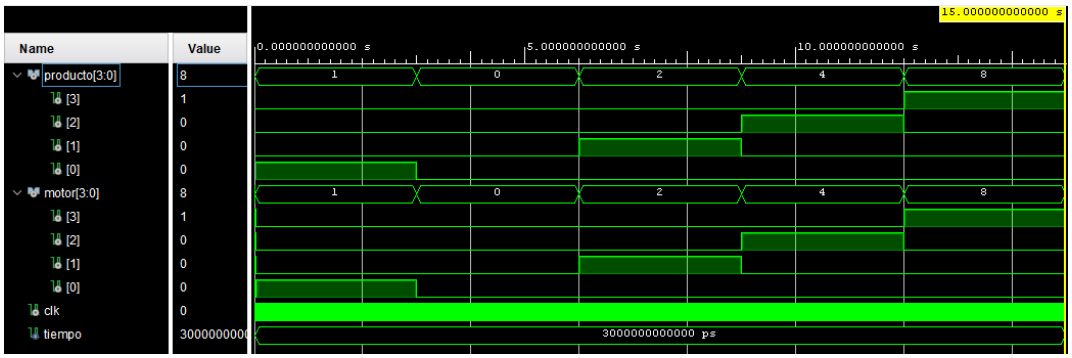
1) **contador_ms > esperaS0ms and flagS0 = ‘1’**
2) **monedero_in = "01100100"**
3) **monedero_in > "01100100"**
4) **contador_ms>esperaS2ms and flagS2='1'**
5) **contador_ms>esperaS3ms and flagS3='1'**

Este diagrama es un esquema del módulo FSM para más información consultar el archivo FSM.vhd del trabajo.

3.3 Diagramas testbench

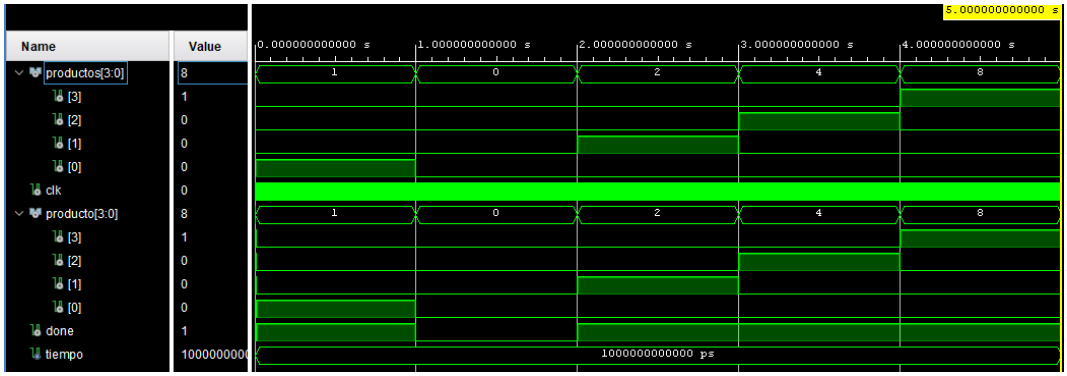
- **motor_salida_tb**

Como se observa en la imagen la llega de un producto del modulo fsm se muestra directamente en motor que es la salida formada por los leds de la placa.



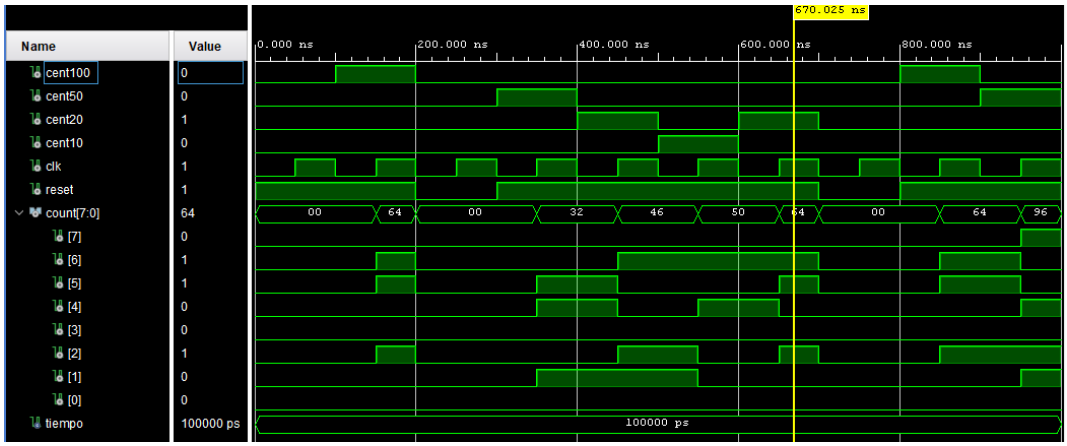
- **Selector_producto_tb**

Se muestra como el producto seleccionando mediante los switches se le asigna a la variable de salida.



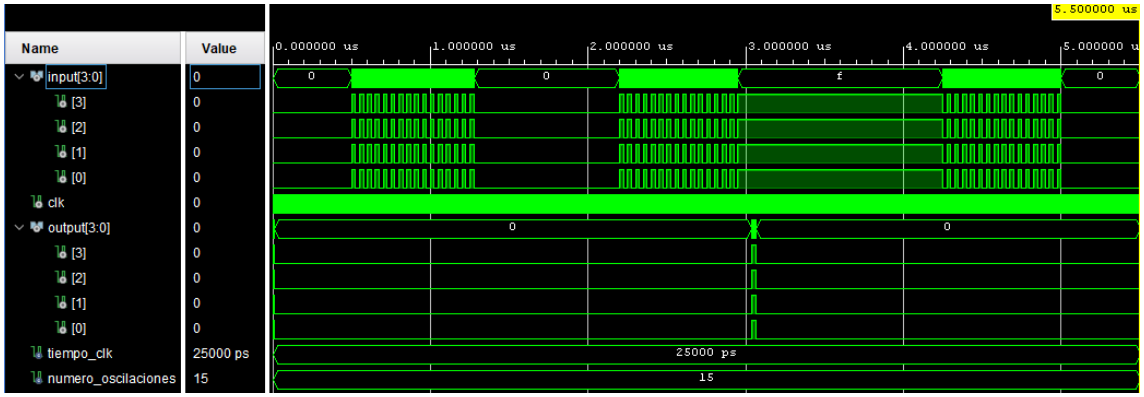
- **monedero_tb**

Se comprueba la suma correcta de las monedas de tal modo que en count al llegar al importe de 100 debe aparecer el numero 100 en binario. El ejemplo se muestra como metiendo una moneda de 50, otra de 20, otra de 10 y otra de 20 la suma es 100. En cambio, también muestra si se pasa de la cantidad como en el tiempo que sigue a 900 ns.



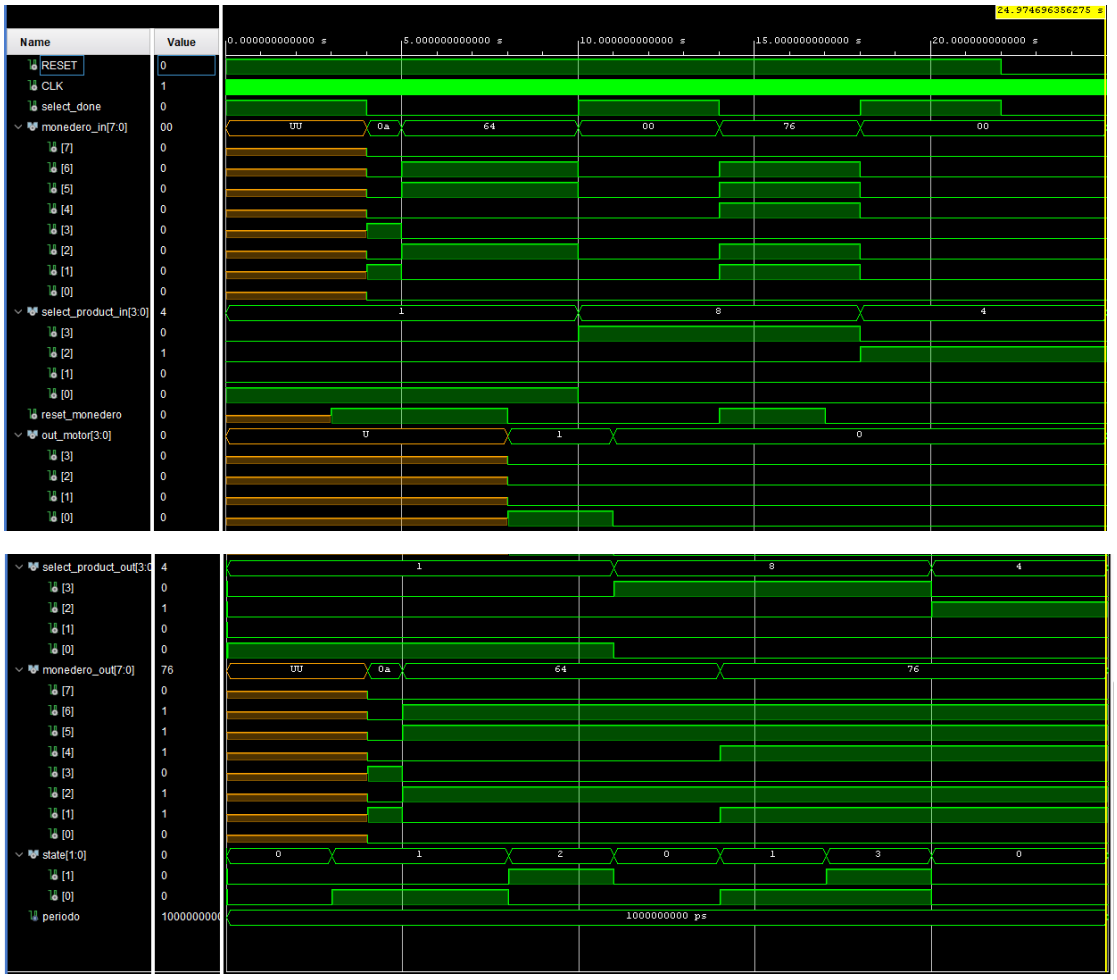
➢ **tratamientoseniales_tb**

Cuando pasa de 0 a una señal de entrada con ruido no devuelve nada, en cambio si pasa de 0 a 1 a la señal de salida se le asigna 1 lógico. Para el caso que hubiera ruido cuando está en uno y termina en '1' se produciría un pulso de 1 por la codificación del detector de flancos.



➢ **FSM_tb**

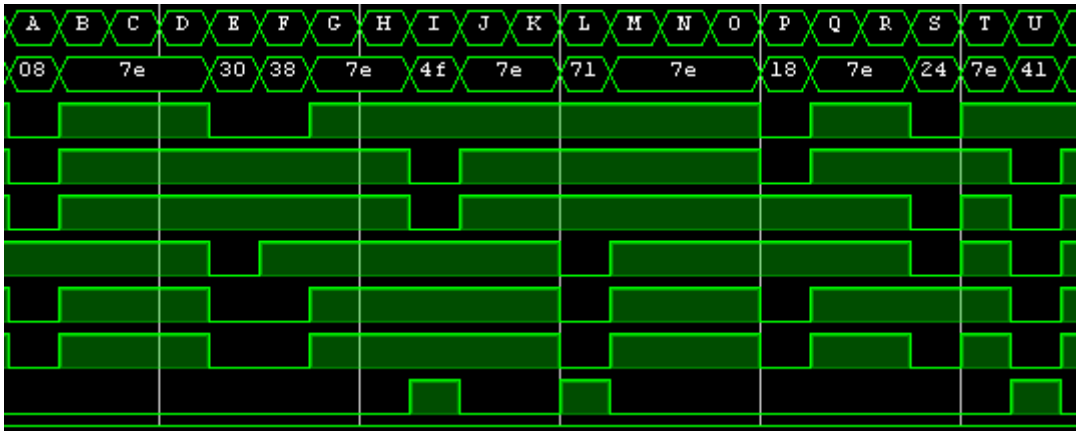
Se realiza una prueba de la selección del producto uno y la simulación de meter la cantidad 100 cts. Se observa como avanza por los diferentes estados S0 S1 y S2 y como mandan los datos a los módulos que están conectados a la FSM como el motor y el display.

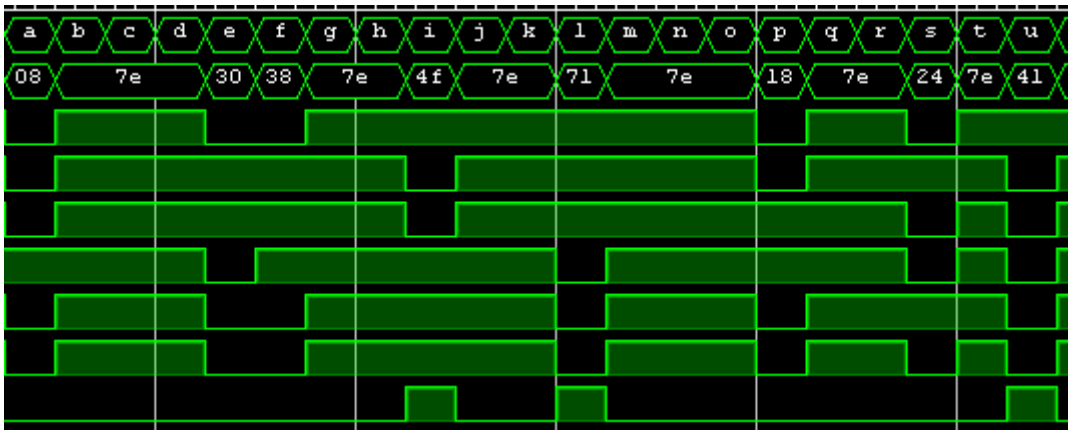


- **decoder_tb**

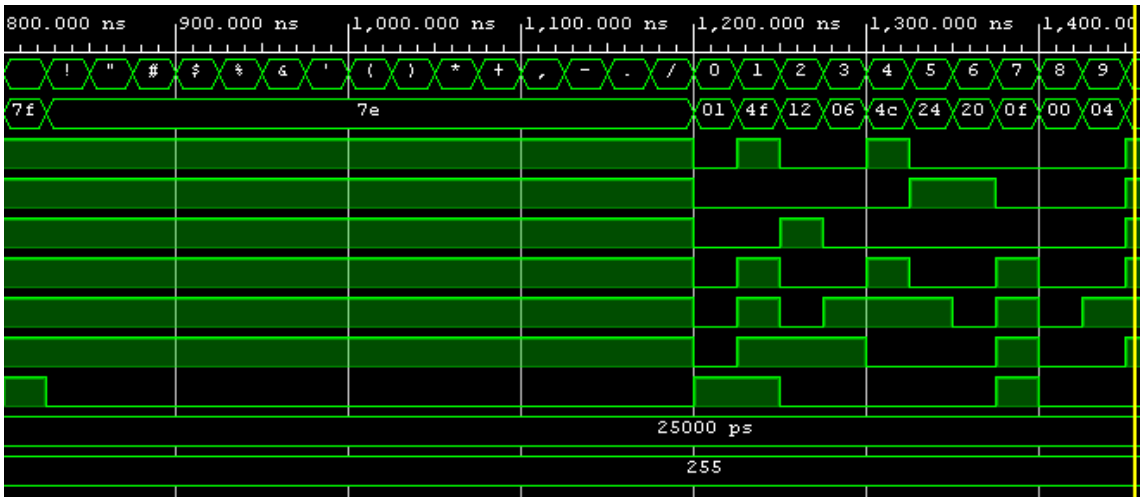
El Código decodificado para cada carácter y número es el de las siguientes imágenes

Para las letras:





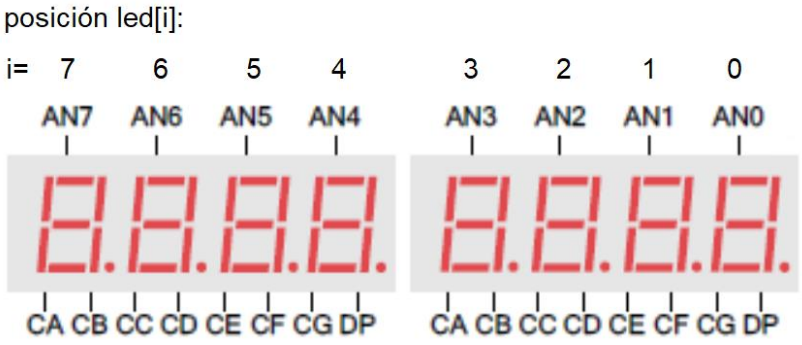
Para los números y el espacio:



Como se ve en la última imagen, si se pone un carácter que no sea una letra, un número o un espacio, el símbolo que sale es el de un guion ‘-‘.

➢ **display_tb**

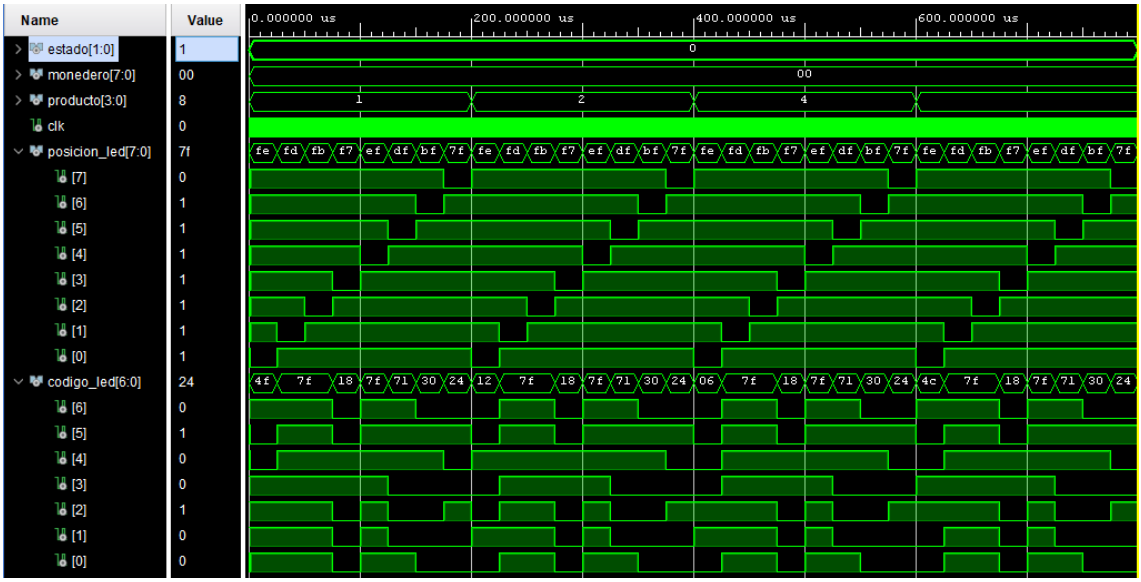
El display que se enciende en cada momento depende del vector posición_led, cada dígito del vector está conectado a los displays según la siguiente imagen:



La posición del led que se enciende cambia cada 1000 pulsos del reloj por defecto para que la tasa de refresco de las letras quede de un 1KHz para que la placa pueda mostrar nítidamente las letras.

Estado 00

La simulación para este estado es el siguiente fragmento:

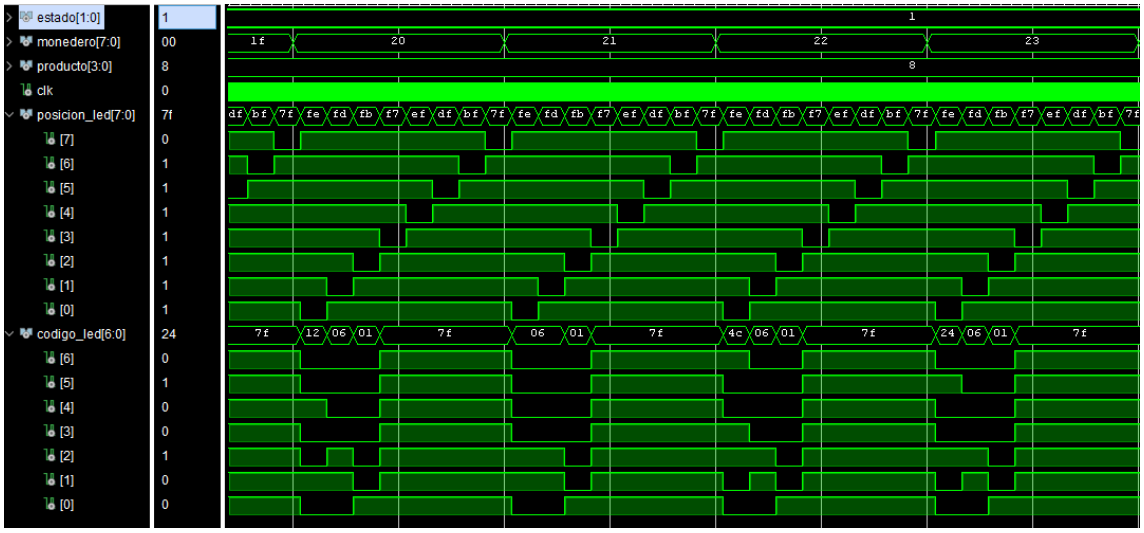


Para este estado se han probado los 4 valores posibles de la entrada producto, “0001” “0010” “0100” “1000” que corresponden a los valores hexadecimales 1,2,4 y 8 respectivamente como se muestra en la simulación, aunque para nosotros los productos están numerados de forma natural ascendente.

En la simulación se ve como escribe en la posición 0 (la del display más a la derecha), el número 1, 2,3 y 4 cuando se selecciona cada producto (consultar el test bench del decodificador para comprobar el código). Además, en las posiciones del 7 al 4 está escrito el mensaje “SEL P” en todo momento.

Estado 01

En este estado se han probado un número arbitrario (150) de números para ver si se escriben correctamente. La imagen muestra unos cuantos de estos números (del 32 al 35):

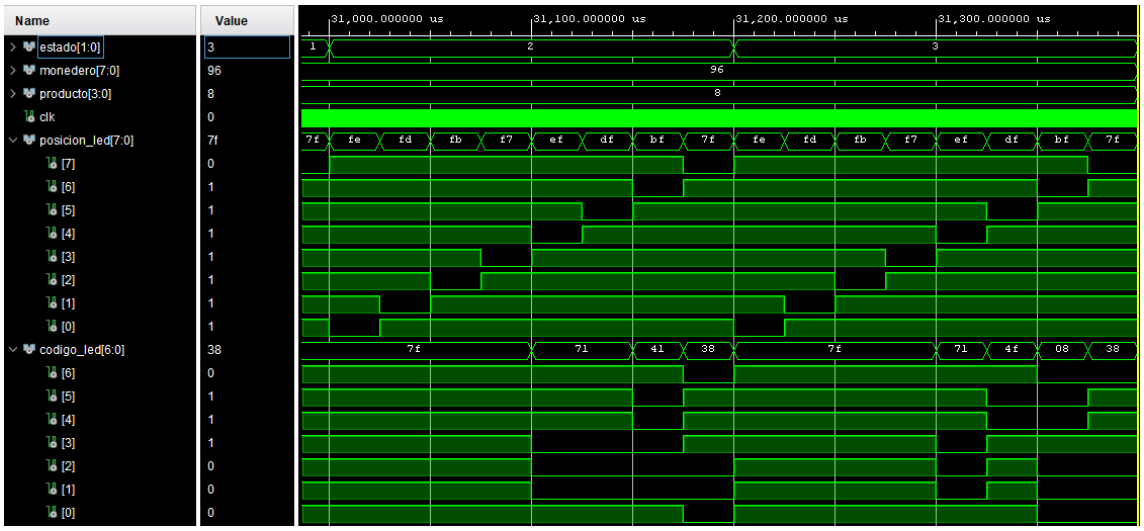


El código que se tiene que ver son 3 números en los 3 últimos displays correspondientes a las unidades, decenas y centenas de la cantidad introducida.

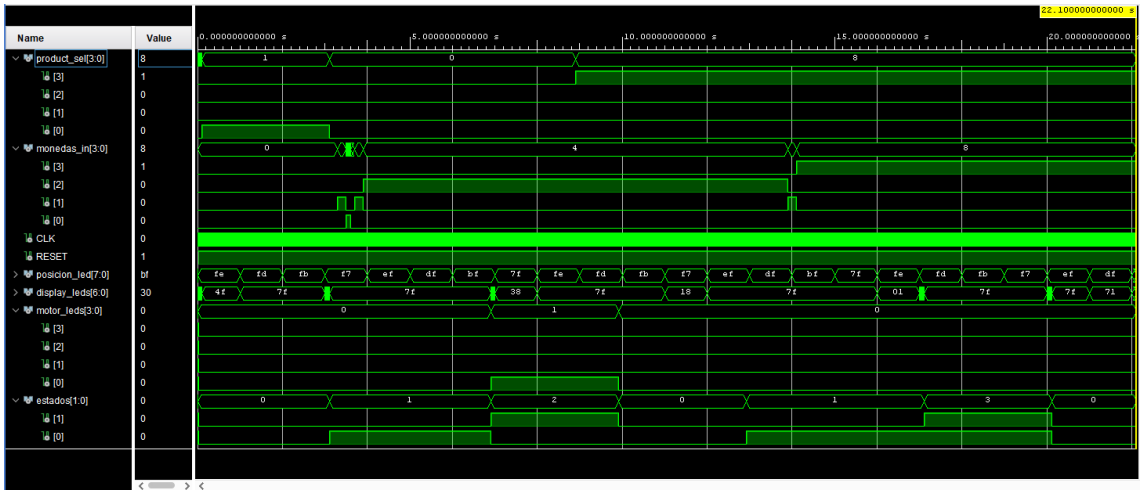
Para los tres casos se ve como el dígito de las centenas (posición_led[2]) está mostrando un ‘0’ (todos encendidos (a ‘0’)menos el del centro (a ‘1’)) y el de las decenas un ‘3’, y el de las unidades va variando con cada número.

Estado 10 y 11

En estos estados tiene que escribir en los dispays de la izquierda el mensaje “FULL” o “FAILL”, dependiendo del estado, todo el rato como se ve en la imagen:



➢ **TOP_tb**



4. EXPLICACIÓN DEL FUNCIONAMIENTO DE LOS BLOQUES FUNCIONALES Y DE SU INTERFAZ

SELECTOR_PRODUCTO

Este bloque recibe como entradas el reloj y un vector de cuatro dígitos, en el cual está a 1 la posición que se corresponde con el switch del producto seleccionado.

Como salidas, salen de este bloque un vector correspondiente al producto seleccionado, de tal modo que, si se selecciona un producto, esta salida se corresponde con el vector de entrada. Si no se selecciona ningún producto la salida tendrá todos sus dígitos a cero. Además, posee una señal que indica que ha sido seleccionado el producto.

```
entity Selector_producto is
    Port (
        productos : in STD_LOGIC_VECTOR (3 downto 0);              -- Entrada del producto seleccionada mediante el switch
        clk       : in std_logic;                                  -- Reloj que gobierna el funcionamiento del modulo
        producto  : out STD_LOGIC_VECTOR (3 downto 0);             -- Salida del producto seleccionado
        done      : out std_logic                                  -- Salida de control para la fsm
    );
end Selector_producto;
```

DECODIFICADOR

Para la visualización se emplean los displays de siete segmentos, por lo que necesitamos este bloque, que se encarga de recibir el número o letra que se quiere visualizar y lo codifica en un vector de siete dígitos, que contiene los leds de los displays que se deben encender para mostrar ese caracter.

```
entity decodificador is
    Port (
        caracter : character;                           -- carácter que se va a decodificar
        led_display : out STD_LOGIC_VECTOR (6 downto 0));-- salida con el código para el display
end decodificador;
```

MONEDERO

Este bloque es un contador síncrono con reset asíncrono. Sus entradas son la señal de reloj, el reset y las entradas correspondientes a las monedas. De este bloque sale un vector con el valor de la cuenta que lleva el monedero según las monedas introducidas.

```
entity monedero is
    port(
        clk : in std_logic; --entrada de reloj
        reset : in std_logic; --entrada de RESET
        cent100 : in std_logic; --entrada moneda de 1 euro
        cent50 : in std_logic; --entrada moneda 50 céntimos
        cent20 : in std_logic; --entrada moneda 20 céntimos
        cent10 : in std_logic; --entrada moneda 10 céntimos
        count: out std_logic_vector(7 downto 0) --salida valor de la cuenta
    );
end monedero;
```

MOTOR_SALIDA

A este bloque entran un vector de cuatro dígitos, en el cual está a 1 la posición que se corresponde con el switch del producto seleccionado y la señal de reloj. Y sale el vector de cuatro dígitos del motor que se encargará de encender el led correspondiente al producto de la entrada.

```
entity motor_salida is
    Port (
            producto : in STD_LOGIC_VECTOR (3 downto 0);          -- LLegada del producto seleccionado
            clk      : in std_logic;                              -- Reloj que gobierna el funcionamiento del modulo
            motor    : out STD_LOGIC_VECTOR (3 downto 0)          -- Salida por los leds
          );
end motor_salida;
```

TRATAMIENTOSEÑALES

Este módulo es el encargado de acondicionar las entradas asíncronas del sistema mediante el sincronizador y el detector de flancos, para que sean utilizadas en la máquina de forma eficiente.

Las entradas son la señal de reloj y el vector de pines de entrada, y las salidas el vector de las salidas.

```
entity tratamientoseniales is
    generic(
        inputsize: integer:=1;                                  -- numero de entradas en el vector de entradas
        sensibilidad: integer:=3);                              -- cuantos '1' seguidos tienen que ocurrir despues de un 0 para que se considere la señal estable
    Port (
        input   : in STD_LOGIC_VECTOR (inputsize-1 downto 0);   -- vector de pines de entrada
        output  : out STD_LOGIC_VECTOR (inputsize-1 downto 0);  -- vector de salidas tratadas
        clk     : in STD_LOGIC);                                -- reloj que gobierna el módulo
end tratamientoseniales;
```

Para implementarlo se usan dos componentes: el sincronizador y el detector de flancos, que se instancian tantas veces como entradas posee el vector de entradas, ya que ambos disponen de una entrada y una salida de un único bit.

SINCRONIZADOR

Este módulo tiene dos entradas, la señal de reloj y la entrada asíncrona al sistema a través de los botones de las monedas.

Se encarga de sincronizar la entrada, de tal modo que se transfiere a la salida si hay un flanco ascendente de reloj.

```
entity sincronizador is
    Port (
        input   : in STD_LOGIC;          -- señal de entrada al módulo
        output  : out STD_LOGIC;         -- señal de salida sincronizada
        clk     : in STD_LOGIC);         -- reloj que gobierna el módulo
end sincronizador;
```

DETECTORFLANCOS

Este módulo tiene dos entradas, que son la señal de reloj y la entrada que sale del sincronizador, y una salida, que es la señal finalmente tratada.

Se encarga de tomar la señal sincronizada y devuelver un pulso a la salida si detecta un flanco positivo.

Además, este bloque posee un filtrado de ruido, que funciona mediante el genérico sensibilidad, que indica cuantos '1' seguidos tienen que ocurrir después de un 0 para que se considere la señal estable.

```
entity detectorflancos is
    generic(
        sensibilidad:integer :=3);                  -- cuantas muestras se toman para verificar que la señal es estable
    Port (
        input   : in STD_LOGIC;                     -- señal de entrada
        output  : out STD_LOGIC;                    -- señal de salida tratada
        clk     : in STD_LOGIC);                    -- reloj que gobierna el módulo
end detectorflancos;
```

DISPLAY

Este módulo gestiona los displays de siete segmentos, es decir la parte de visualización de la máquina expendedora.

En primer lugar, dispone de cuatro entradas, la entrada de reloj, el valor de la cuenta del monedero, el producto seleccionado y el estado en el que se encuantra la máquina de estados.

En segundo lugar, dispone de dos salidas, una con el código para mostrar el carácter correspondiente y otra con la posición del display en el que vamos a escribir el carácter.

Según el estado que reciba, introduce en los displays unos mensajes u otros.

```
entity display is
    generic(prescaler:integer:=1000);                   -- sirve para modificar la velocidad de rotación del display que se va a escribir
    Port ( estado :     in STD_LOGIC_VECTOR (1 downto 0);   -- estado del fsm que decide que mensaje se va a escribir
           monedero :   in STD_LOGIC_VECTOR (7 downto 0);   -- valores del contador de dinero para escribir en el estado "01"
           producto :   in STD_LOGIC_VECTOR (3 downto 0);   -- producto seleccionado para escribir en el estado "00"
           clk :        in STD_LOGIC;                       -- reloj para sincronizar los displays con el programa
           posicion_led:out STD_LOGIC_VECTOR (7 downto 0);  -- vector de salida que devuelve el led que se debe encender en cada momento
           codigo_led:  out STD_LOGIC_VECTOR(6 downto 0));  -- vector de salida con el código para encender los leds de los display para formar una letra o número
end display;
```

FSM

Este módulo gestiona los estados por los que pasa la máquina. Además, recibe datos de otros módulos y según las entradas y el estado envía datos a otros módulos.

En primer lugar, tiene cinco genéricos, uno para la frecuencia, ya que este módulo funciona a distinto ritmo y adapta la señal de reloj que recibe, y cuatro que se corresponden con los tiempos de transición entre estados.

En segundo lugar, dispone de cinco entradas, la de reset, la de la señal de reloj, la de la cuenta de dinero, que procede del monedero, la del producto seleccionado y la que indica que se ha seleccionado, que provienen del selector.

En tercer lugar, dispone de cinco salidas, la que se encarga de resetear el monedero si la cuenta es igual o superior a un euro, la que irá al motor para indicarle el producto, la que va al display para mostrar el producto seleccionado, la que va al display para mostrar la cuenta y otra que envía el estado al display para su gestión interna.

Los estados son los cuatro que veíamos en el diagrama de estados del apartado 3.2.

La máquina parte del estado de reposo S0, en el que recibe los datos del selector de productos. Si se ha seleccionado el producto pasa al estado S1, donde se compara la cuenta que recibe del monedero. Si llega a 100 pasa al estado S2 y si pasa de 100 va al estado S3. En el estado S2 manda el código del producto al motor y en el S3 entra en la fase de error. Tras el estado S2 o el S3 vuelve al estado de reposo y se resetea el monedero.

```
entity FSM is
    generic(
        frecuenciarelojKHz:integer:=100000;                    -- frecuencia a la que funciona el reloj del sistema
        esperaS0ms:integer:=3000;                              -- tiempo de espera para pasar al siguiente estado
        esperaS1ms:integer:=3000;
        esperaS2ms:integer:=3000;
        esperaS3ms:integer:=3000);
    Port (
        RESET              : in std_logic ;                    -- reset activo a nivel bajo
        CLK                : in std_logic ;                    -- reloj que gobierna el funcionamiento del módulo
        monedero_in        : in std_logic_vector (7 downto 0); -- entrada del contador de dinero del módulo monedero
        select_product_in  : in std_logic_vector (3 downto 0); -- entrada del producto seleccionado
        select_done        : in std_logic ;                    -- indicador de que el producto ha sido seleccionado
        reset_monedero     : out std_logic;                    -- reseteo del contador de monedero
        out_motor          : out std_logic_vector (3 downto 0); -- señal de activación del led según el producto seleccionado
        select_product_out : out std_logic_vector (3 downto 0); -- señal que va al display con el producto seleccionado
        monedero_out       : out std_logic_vector (7 downto 0); -- señal que va al display con el dinero insertado
        state              : out std_logic_vector (1 downto 0)); -- señal que va al display con el estado actual
end FSM;
```

TOP

Este módulo es el encargado integrar todos los módulos anteriores dando lugar a nuestro sistema final. Sus entradas son las entradas al sistema: la selección del producto, que viene determinada por los interruptores, las monedas, que se simulan mediante los botones de la placa, la señal de reloj de la placa y el reset de la placa.

Por otra parte, sus salidas son la posición del display donde se va a mostrar la letra o número del mensaje a transmitir, el código para dibujar el caracter en el display de siete segmentos, la salida del motor correspondiente al producto seleccionado, que se simula activando el led del producto y el estado actual en el que se encuentra el sistema, que será empleado para la simulación del bloque.

Además, este bloque dispone de las señales necesarias para realizar la comunicación interna entre los distintos módulos del sistema.

```
entity TOP is
    generic(frecuenciarelojKhz:integer:=100000);
    Port (
            product_sel  : in std_logic_vector(3 downto 0);                -- Entrada del producto seleccionado con los switches del 0 al 3
            monedas_in   : in std_logic_vector(3 downto 0);                -- Entrada que simula las monedas introducidas mediante botones
            CLK          : in std_logic;                                   -- Reloj que contola el sistema
            RESET        : in std_logic;                                   -- Activo a nivel bajo resetea la maquina
            posicion_led : out std_logic_vector (7 downto 0);              -- Posición del display en la que se va a mostrar el dígito o la letra
            display_leds : out std_logic_vector (6 downto 0);              -- Contiene el codigo para dibujar el dígito o la letra
            motor_leds   : out std_logic_vector (3 downto 0);              -- Salida que simula mediante leds el proceso en el que la maquina saca el producto
            estados      : out std_logic_vector(1 downto 0)                -- salida del estado actual utilizada para el testbench
    );
end TOP;
```