PROYECTO SISTEMAS ELECTRÓNICOS DIGITALES

MEMORIA TRABAJO VHDL

MÁQUINA EXPENDEDORA EN FPGA

NATALIA BORLAF NIETO, 54516

JESÚS GARCÍA SÁNCHEZ, 54622

ISMAEL FERNÁNDEZ DE LA COTERA LORENZO, 54594

CURSO 2022-2023

Repositorio github:

https://github.com/nborlaf/Trabajo-SED---VHDL.git

ÍNDICE

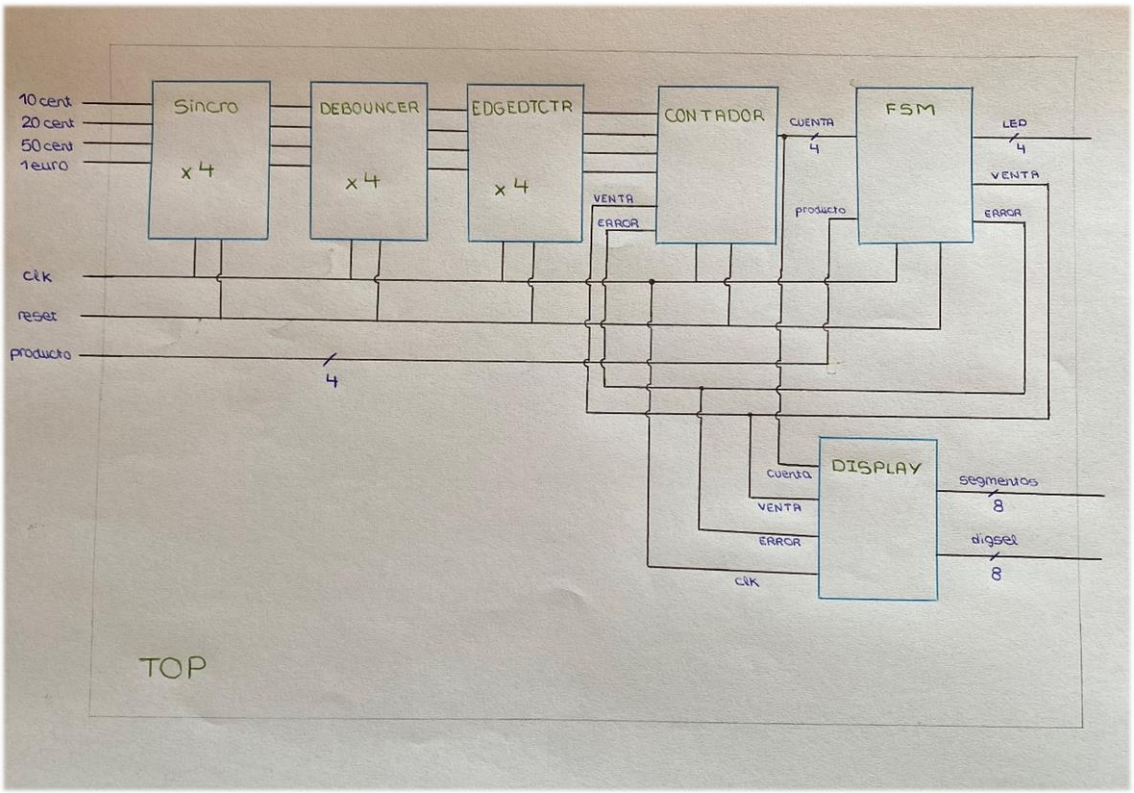
INTRODUCCIÓN

El proyecto presentado de VHDL se implementa en una FPGA (Nexys-4-DDR-Master) simulando una máquina expendedora. El usuario podrá elegir entre 4 productos diferentes con un precio de 1.2 euros.
La máquina permitirá seleccionar el producto a través de los interruptores, introducir el dinero pulsando los botones y así venderá el producto. Gracias al *display* se podrá ver la cuenta del dinero y un mensaje cuando se produzca un error o cuando se venda el producto.
Los botones incluidos que hacen referencia a las monedas tienen unos valores de 10, 20 y 50 céntimos, y un euro, para poder alcanzar la suma de 1.2 euros de los productos.
Cuando el usuario elige un producto lo hace a través de los *switches* de manera que se enciende el led correspondiente al producto para indicarlo.
El programa empezará en un estado con la cuenta a 0. Primero se elegirá el producto, después se ingresa el dinero sin pasarse de la cantidad indicada, y después se venderá el producto esperando un tiempo que simula la entrega del producto.
Todas las opciones que se salgan de los pasos indicados están contempladas y explicadas posteriormente, teniendo así un modo de error para todas estas incidencias.
También existe un botón de *reset* que pone los valores de los registros a 0 y vuelve al estado inicial. Este botón es activo a nivel bajo.

ENTIDADES

TOP

La entidad TOP es la entidad principal, en la que se recogen todos los componentes de las demás entidades que componen el programa.
El trabajo se compone de varias entradas para su buen funcionamiento: 4 entradas para los botones de 10cent, 20cent, 50cent y 1euro, la señal de reloj que llevará el sistema, la señal de *reset*, y un bus de 4 bits para cada producto que corresponde con los *switches*.
Como salidas tendrá un bus de 4 bits para cada led según el producto que se haya elegido, y dos salidas de buses de 8 bits para el *display*. Estas dos últimas salidas se usarán para encender cada uno de los *displays* (digsel) y para dentro de cada *display* elegir que segmentos se encenderán (segmentos).
Para el acondicionamiento y sincronización del reloj con las señales de los botones se utilizarán las entidades Sincro, DEBOUNCER Y EDGEDETCTR. Como hay 4 botones para las diferentes monedas, se necesitarán 4 componentes de cada una de estas entidades.



Para su implementación se ha utilizado un for-generate. Para una mayor comodidad para usar este recurso, se ha creado un vector monedas que agrupa las señales de cada uno de los botones.

SINCRO

Sincro corresponde con el sincronizador del proyecto que se encarga de sincronizar las señales de entrada de los botones con el reloj de la placa. Gracias al sincronizador se evita la metaestabilidad cuando hay un cambio en la señal de entrada durante un flanco de reloj.
Se utilizarán tantos sincronizadores como botones de entrada, que simbolizan las monedas, haya. En este caso, se utilizarán 4 sincronizadores, una por botón de entrada. También tiene entrada para el reloj y *reset*, poniendo a 0 las variables.

DEBOUNCER

El debouncer se necesita para evitar los rebotes que se producen al pulsar los botones. En esta entidad se compara el estado anterior con el actual durante un periodo de tiempo para ver si los valores concuerdan y se obtiene un valor estable. Si se obtiene ese valor estable es cuando se considera que se ha pulsado el botón. Al igual que en el sincronizador se repetirá la estructura 4 veces, una por botón.
Además, cuenta con entrada para el reloj y *reset*.

EDGEDTCTR

La pulsación mecánica de los botones tiene una duración aleatoria, dada por el tiempo que el usuario mantiene el botón pulsado. Esto puede llevar a inconvenientes a la hora de registrar la señal introducida. Este módulo se emplea para que cada vez que se produzca un flanco de bajada en el pulsador, la señal de salida del módulo se active, contando así solamente los flancos del pulsador, evitando así que se produzcan cambios continuamente mientras el botón se encuentra pulsado.
Al igual que en el sincronizador se repetirá la estructura 4 veces, una por botón.
Además, cuenta con entrada para el reloj y *reset*.

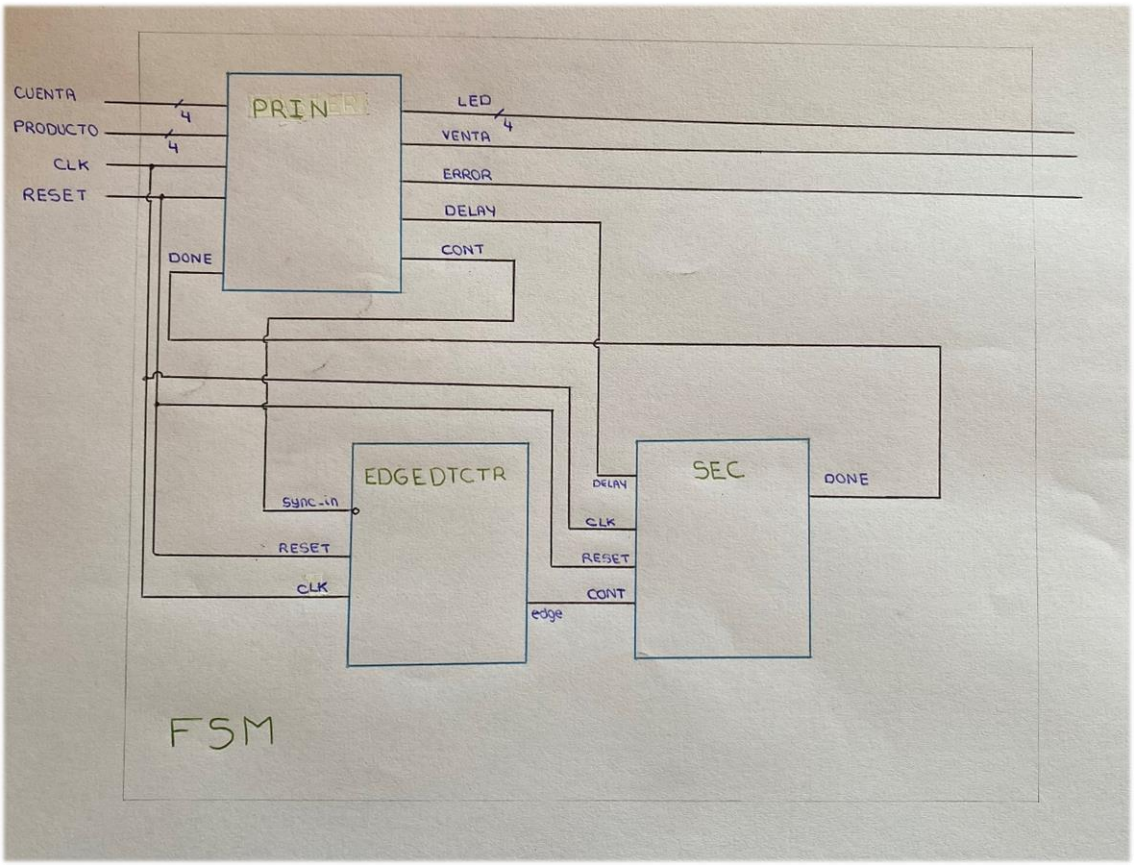
CONTADOR

La entidad CONTADOR se encarga de llevar la cuenta del dinero introducido por los botones.
En esta entidad se incluirá un registro, cuenta_aux, de 5 bits que llevará la cuenta. Al pulsar el botón de los 10 cent se añadirá una unidad, al pulsar 20 cent se añadirán 2 unidades, con 50 cent se añadirán 5 y con un euro se añadirán 10 unidades. Todo esto se realizará de forma asíncrona.
Este registro tiene 5 bits, uno más que la salida, para evitar el desbordamiento a la hora de realizar las sumas. Si llega a pasar, como sucede cuando sumas un euro más un euro (10100), la entrada error se activaría dejando cuenta_aux y cuenta a 0 para poder seguir sumando.
La entidad cuenta con entradas de ERROR y VENTA para detectar un error o para indicar que un producto se está vendiendo. En cualquiera de estos dos casos, si alguna de estas entradas se activa, la cuenta pasará a valer 0 de manera síncrona.
El resultado obtenido en cuenta_aux será transmitido a la salida cuenta según las siguientes equivalencias:

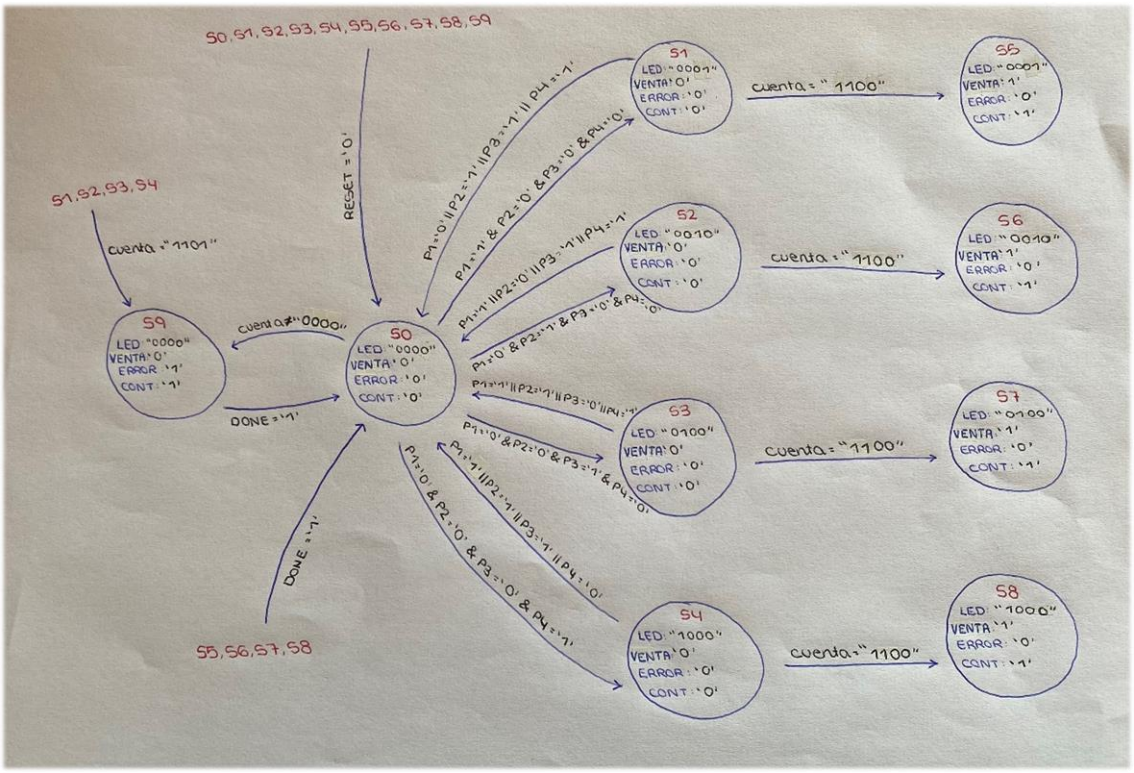
DINERO	Cuenta_aux	CUENTA
0€	00000	0000
0.1€	00001	0001
0.2€	00010	0010
0.3€	00011	0011
0.4€	00100	0100
0.5€	00101	0101
0.6€	00110	0110
0.7€	00111	0111
0.8€	01000	1000
0.9€	01001	1001
1€	01010	1010
1.1€	01011	1011
1.2€	01100	1100
>1.2€	Otros valores	1101

FSM

Esta entidad será la encargada del funcionamiento de la máquina de estados. La entidad FSM se compone de dos submáquinas (maestra y esclava) para realizar tareas de temporización.



La submáquina FSM_PRIN será la encargada de realizar los cambios de estado y los cambios de las señales de salida de FSM.
La submáquina FSM_SEC se encargará de las tareas de temporización. Se iniciará el temporizador con el tiempo que le pase FSM_PRIN en la salida *delay* cuando detecte un flanco ascendente de la señal *cont*. Cuando la cuenta termine, se enviará a FSM_PRIN un 1 lógico en la señal *done* para indicárselo.
En la siguiente imagen se puede ver el diagrama de estados.



El estado de inicio será S0, en el que todas las salidas de la máquina se encontrarán a nivel lógico bajo.

Una vez el usuario active un *switch* indicando que desea alguno de los productos pasará al estado S1, S2, S3 o S4, dependiendo del producto elegido, y se encenderá el led correspondiente a ese producto. En caso de que se apague ese *switch* o se active algún otro a la vez se volverá al estado de reposo S0.

Estando con un producto seleccionado se tendrá que introducir la cantidad de 1.2 euros para pasar a los estados de venta S5, S6, S7 y S8, en los que se activará *venta* y *cont* cuando haya pasado el tiempo indicado para la compra. Después de la venta se volverá al estado inicial S0.

Si introduciendo el dinero se sobrepasa la cantidad de 1.2 euros se cambiará al estado S9 donde se activa la señal de error y se resetea la cuenta. Después del tiempo indicado para la devolución del dinero se pasará al estado S0.

Existe la opción de que estando en S0 sin ningún producto seleccionado se introduzca dinero, de tal manera que se pasará al estado de error S9 explicado anteriormente.

Si se activa la señal de *reset* (a nivel lógico bajo), estando en cualquiera de los estados, se volverá al estado inicial S0.

FSM_PRIN

Esta entidad se compone de 3 procesos diferentes:
- ACTUALIZACION_DE_ESTADO: Este proceso se encarga de a la hora de recibir la señal de reset pasar al estado S0 de forma asíncrona, teniendo así prioridad. También se encarga de, en el flanco de reloj, cambiar del estado actual al siguiente.

- CAMBIO_DE_ESTADO: Según el estado en el que se encuentre asignará el siguiente estado al correspondiente indicado en el diagrama de estados. Esto se realizará según se cumplan las condiciones de *cuenta*, *producto* y *done*.

- ACTUALIZACION_SALIDAS: Se encarga de asignar los valores de salida que vienen en el diagrama dependiendo del estado en el que se encuentren.

Para el uso de las señales *present_state*, correspondiente al estado actual, y *next_state*, que representa al estado al que se quiere pasar, se crea un tipo de datos llamado STATE. La señal *present_state* se inicializa en S0 por ser el estado de inicio.

```
type STATE is (
    S0, --ESTADO DE REPOSO
    S1, --PRODUCTO 1 SELECCIONADO
    S2, --PRODUCTO 2 SELECCIONADO
    S3, --PRODUCTO 3 SELECCIONADO
    S4, --PRODUCTO 4 SELECCIONADO
    S5, --VENDIENDO PRODUCTO 1
    S6, --VENDIENDO PRODUCTO 2
    S7, --VENDIENDO PRODUCTO 3
    S8, --VENDIENDO PRODUCTO 4
    S9  --ESTADO DE ERROR
    );
signal next_state : STATE;          -- Estado siguiente
signal present_state : STATE:=S0;   -- Estado actual
```

La entidad cuenta también con las constantes DURACION_ERROR y DURACION_VENTA para los tiempos que se han contado anteriormente, que representan el tiempo que tarda en devolver el dinero en el estado S9 y el tiempo que tarda en sacar el producto tras su venta.

```
constant DURACION_ERROR : positive :=400000000; --DURACIÓN DE ESPERA TRAS ESTADO DE ERROR EN CICLOS DE RELOJ
constant DURACION_VENTA: positive :=600000000; --DURACIÓN DE ESPERA TRAS VENTA
```

FSM_SEC

Esta entidad se encarga de llevar la cuenta de los temporizadores.
Para que la detección de flancos funcione debidamente, se ha añadido un EDGEDTCTR con puerta *not* de entrada para la salida *cont* de FSM_PRIN que corresponde con la entrada *cont* de FSM_SEC. Esto se hace para que se detecten los flancos negativos.
Una vez se detecta el flanco ascendente en *cont* se carga la señal *count* con el tiempo de *delay*, *aux_done* a '0' y *aux_cont* a '1'. En este momento el temporizador empezará la cuenta.
Cuando el temporizador acabe se activará *aux_done* que es el encargado de controlar la salida DONE. Al activarse *aux_done* indicando la finalización del temporizador se desactivará *aux_cont*. Si *aux_cont* ya estaba desactivada antes de terminar el temporizador, *aux_done* no se activará, indicando que no se había mandado al temporizador hacer la cuenta.

DISPLAY

La entidad Display es la encargada de mostrar en los displays el dinero que se ha introducido en la máquina y los mensajes de sold y fail. Cuenta con una entrada de 4 bits que indica el dinero que se ha introducido(cuenta), una entrada de 1 bit (venta), que muestra la palabra "sold" en los displays cuando es '1', otra entrada de de 1 bit (error) que muestra la palabra "fail" en los displays cuando es '1', y una ultima entrada para el reloj (clk). También cuenta con 2 salidas: la salida digsel de 8 bits que indica cuáles de los 8 displays de la placa se van a encender, y la salida segmentos de 8 bits (7segmentos + el punto) que indica qué segmentos de el digito se van a encender. Debido a que los cátodos de los segmentos son comunes a los ocho displays, se mostraría el mismo digito en cada display. Para que en los displays no muestren lo mismo, los displays se manejarán en periodos de 1.6ms, es decir, como tenemos 8 displays, cada display se ilumina durante 1/8 periodo de forma consecutiva. Por ejemplo, si se quiere mostrar en los displays el número 71, lo que se haría es encender primero el digito "1" durante 1/8 periodo (0.2ms), transcurrido ese tiempo, se enciende el digito "7" durante otros 1/8 periodo, y esto se repetiría cada periodo. Como se repite lo mismo cada 1.6ms, se verá que el número 71 estará encendido continuamente en los displays. Para implementarlo en el código se ha utilizado un signal clk_counter que cuenta el número de flanco de subida del reloj (clk), como el reloj es de 100MHz, el número de flancos que tiene que contar para un periodo de 1.6ms es 160000, y para 1/8 periodo el clk_counter tiene que contar 20000 flancos de subida, lo que equivale a 0.2ms. Lo que se quiere hacer es que

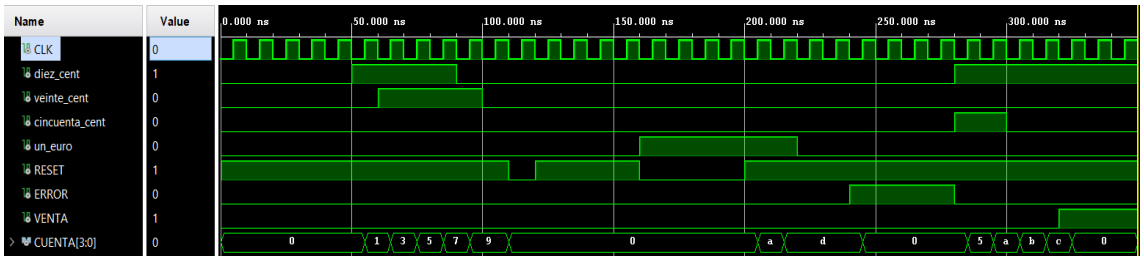
cada vez que se cuente hasta 20000, se incrementa el signal cifra que será el encargado de indicar el display que se quiere encender, por ejemplo, si es 0 se encienda el primer display y si es 1 se enciende el segundo display.

TESTBENCHES

Para comprobar el correcto funcionamiento del programa antes de subirlo a la placa se van a realizar testbenches en las entidades que consideramos que más problemas pueden dar.
Esto nos permitirá visualizar de forma fácil las señales en el diagrama de tiempos y poder subsanar los errores que haya.

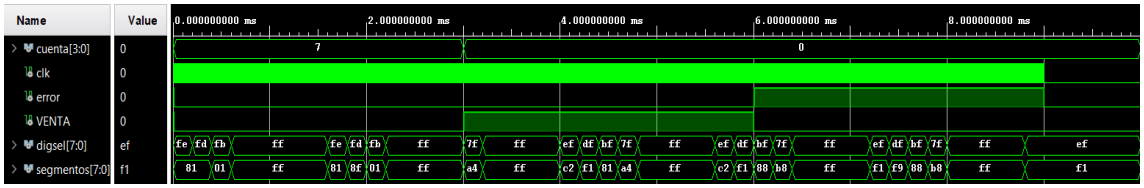
A continuación, se explicarán más detalladamente los testbenches de las entidades que se hayan realizado.

TB_CONTADOR



La entidad contador se ha simulado con un tiempo de 350ns con una señal de reloj de100Mhz. En el diagrama de tiempos se observa que el contador funciona como se espera, añadiendo a la cuenta el valor de la moneda de más valor que este activa y reiniciándose cuando se activan las señales de RESET, VENTA o ERROR.

TB_DISPLAY



La entidad display se ha simulado con un tiempo de 10ms con una señal de reloj de 100Mhz. Como podemos ver en el diagrama de tiempo, la entidad funciona como se espera. La señal segmentos y la señal digisel, las cuales controlan el display, cambian en función de si el valor de cuenta varia (se visualiza el numero correspondiente) o si la señal VENTA o ERROR están activas (se visualiza la palabra correspondiente).