UNIVERSIDAD POLITÉCNICA DE MADRID

escuela técnica superior de ingeniería y diseño industrial

TRABAJO VHDL

MÁQUINA EXPENDEDORA

Ismael Torrijano Pedroche 55492
Asier Miño Sierra 55364
Sergio Cabo Rodríguez 55161

INTRODUCCIÓN

En la actualidad, los dispositivos embebidos tienen una gran presencia en nuestro entorno cotidiano, desde electrodomésticos hasta sistemas de transporte. Estos dispositivos suelen estar basados en microcontroladores o FPGAs, que son capaces de ejecutar programas de control y realizar tareas específicas. Uno de los lenguajes de descripción de hardware más utilizados en la industria y la investigación es VHDL (VHSIC Hardware Description Language), que permite describir sistemas digitales de manera precisa y portátil.

Placa Nexys 4DDR

El objetivo de este trabajo es demostrar la capacidad de VHDL para describir y controlar sistemas digitales complejos, así como la aplicabilidad de la placa Nexys 4DDR en el desarrollo de proyectos de electrónica digital utilizando VHDL.

PLANTEAMIENTO DEL PROBLEMA

Actualmente, las máquinas expendedoras son dispositivos muy comunes en nuestro entorno, ya que nos permiten adquirir productos de manera rápida y sencilla. Sin embargo, estos dispositivos suelen estar basados en microcontroladores o FPGAs de propósito general, lo que puede limitar su rendimiento y flexibilidad.

Por tanto, el problema a abordar en este trabajo es el de desarrollar una máquina expendedora que cumpla con los requisitos de diseño previamente dados y a los que se ha de ceñir el proyecto. Algunos de estos requisitos son: realización de varios testbenches, división del trabajo en entidades, realización de diagramas de bloques/entidades, uso de E/S tales como pulsadores, LEDs o displays, etc.

DISEÑO E IMPLEMENTACIÓN

Funcionamiento básico

Para el diseño de la máquina, se ha tenido en cuenta la elección de producto por parte del usuario antes que la introducción de dinero en la máquina. El funcionamiento comienza cuando el usuario selecciona el producto que quiere de entre los tres posibles mediante un switch. Una vez se ha seleccionado el producto el sistema pasa a la compra de este, y para ello entra en la fsm correspondiente (Producto1, Producto2 o Producto3) donde ahora el usuario deberá introducir el dinero en forma de moneda de una en una y mediante la pulsación de los botones de moneda, los cuales comprenden monedas de 0.5€, 1€ y 2€. Cada vez que el usuario introduce una moneda, el sistema pasa a un estado distinto, donde lleva la cuenta internamente. En el momento en que el usuario introduce una moneda que supone llegar al precio del producto o pasarse, el sistema entra en un estado en el que libera el producto haciendo entrega de este. Si el compendio de monedas introducidas hiciera una cantidad de dinero superior al precio del producto, se entregaría el producto naturalmente, además del cambio correspondiente (0.5€, 1€, 1.5€ o 2€).

Si el usuario lo desea, puede seguir comprando el producto seleccionado mediante la sucesiva introducción de monedas, lo cual haría volver al ciclo de compra y venta.

Si por el contrario el usuario quiere cambiar de producto, debería pulsar el botón de nueva compra (una vez terminada la compra anterior), deseleccionar el producto anterior mediante el switch y seleccionar el nuevo producto a elegir, que tendrá un nuevo precio correspondiente, y el proceso de compra se iniciaría otra vez.

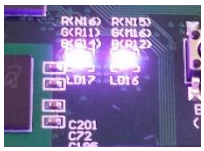
Implementación Hardware

Este trabajo, se ha desarrollado utilizando la placa Nexys 4DDR como plataforma de implementación. Para el cumplimiento de las necesidades de funcionalidad se han utilizado diversas entradas y salidas como switches para la selección del producto; pulsadores para la introducción de monedas al sistema; leds para el aviso de venta de productos y displays de 7 segmentos para devolución de mensajes que informen al usuario del estado de la compra, cambio, selección de producto, etc.

Pulsadores

Display

Switches



LEDs

Implementación Software

La herramienta de descripción de hardware que se ha utilizado ha sido Vivado, donde se han programado diversas entidades que cumplen funciones varias. Todas estas entidades se pueden listar de la siguiente manera:

- TOP
- Sincronizador
- Debouncer
- Edge_detector
- Control de Display y LEDs
- Sel_producto
- FSM
- Producto1
- Producto2
- Producto3

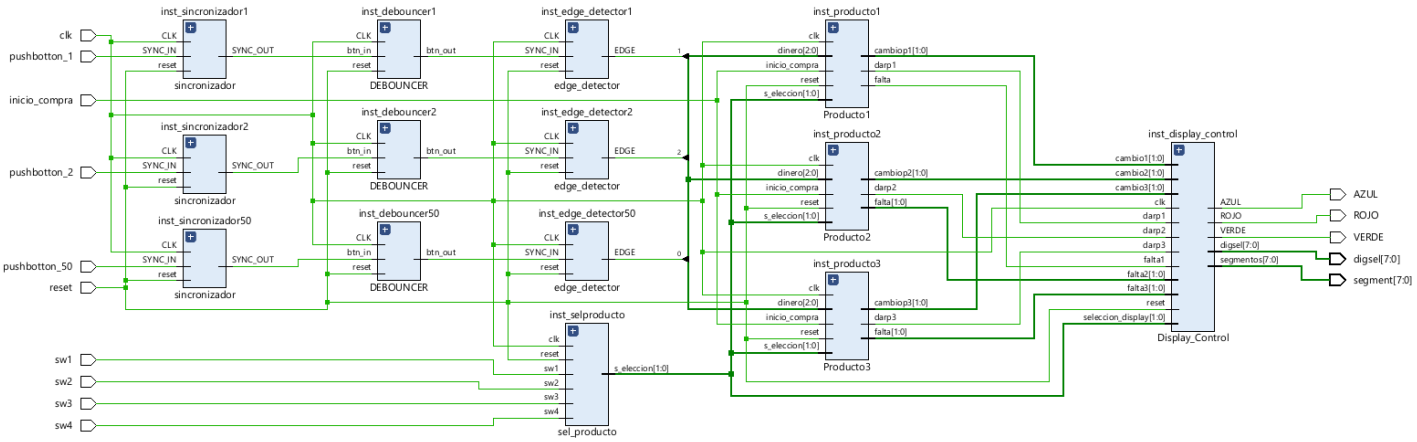


Diagrama del proyecto en VHDL

Top

La entidad top es en la cual se declaran todos los componentes del sistema.

Señales de reloj (se usará una señal de reloj síncrona para los debouncer, sincronizador y Edge_detector) y otro reloj distinto para el display y reset.

LEDs RGB para indicar la compra.

Botones para 50 céntimos, 1 euro y dos euros que luego se trataran como una señal de tres bits en la FSM. También tenemos un botón de inicio de compra el cual una vez estamos en la FSM de un producto si luego nos gustaría comprar otro simplemente lo pulsamos pudiendo asi seleccionar otro.

El filtrado de la señal de las monedas (sincronizador, DEBOUNCER Y EDGE_DETECTOR) se hace de manera individual, pero somos conscientes que podríamos hacer uso de un bucle for que declarase las necesarias evitando tener que hacer cada entidad por separado, en este caso 3.

La selección de producto (sw1, sw2, sw3) la hemos realizado con un vector de dos bits 01 prod1, 10 prod 2, 11 prod3, esta señal nos será muy útil a lo largo del programa como enable tanto de las FSM como del control_display.

La señal de cambio un vector de dos bits que representa las monedas de 0.5e, 1e y 2e que devuelve la máquina.

Señales de dar, se ponen a 1 cuando se realiza la venta del producto.

Señales de falta de dos bits que nos sirven para reflejar en el display el dinero que falta para realizar la compra.

Sincronizador

Se emplearán 3 sincronizadores, uno por cada señal de entrada producida por los botones que simbolizan las diferentes monedas que se introducen en la máquina. El sincronizador se encarga, como su propio nombre indica, en sincronizar las señales de entrada con la señal de reloj de la placa, evitando así la meta estabilidad ocurrida cuando se produce un cambio en la señal de entrada durante un flanco del reloj, pues la pulsación del botón se puede producir de forma asíncrona al reloj. También cuenta con una entrada para el reloj y de reset.

Edge_detector

Se requiere para evitar el efecto de los rebotes que se dan al producir las señales con los botones ya que los pulsadores tienen una naturaleza mecánica. Para ello se utilizan flip flops que son capaces de almacenar el estado anterior, si durante un periodo de tiempo amplio el valor actual concuerda con ese valor almacenado se puede asumir que el estado es estable. Al igual que el sincronizador se repite la estructura 3 veces, una por cada botón.

También cuenta con una entrada para la señal de reloj y otra para la de reset. La realización en cuanto a disposición en la entidad top es la misma que en el caso del sincronizador, instanciamos 4 veces la entidad debouncer.

Debouncer

La pulsación mecánica de los botones no es de una duración determinada, por lo que se pueden encontrar 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 los dos módulos anteriores este también se empleará tres veces, una por cada señal de botón.

También cuenta con una entrada para la señal de reloj y otra para la de reset.

Control de Display y LEDs

El funcionamiento del display para iluminar todos los dígitos se basa ir encendiendo el primer digito, luego el siguiente (mediante el uso de un case) y así hasta llegar a los 8 dígitos (o los que queramos usar) pero a una frecuencia tan alta que es imperceptible para el ojo humano y se hace la ilsuión de que esta luciendo todo el display a la vez.

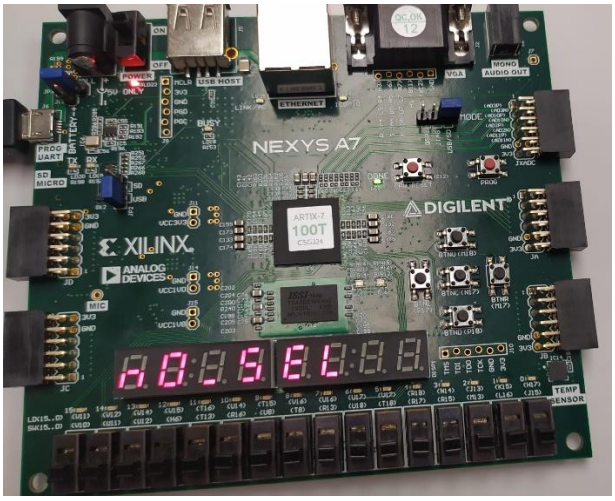
```
process(clk)
begin
    --Periodo 1.6 ms-> clk_counter=160000
    if rising_edge(clk) then
        clk_counter<=clk_counter + 1;
        --periodo/8 = 0.2 ms -> clk_counter=20000
        if clk_counter>=20000 then
            clk_counter<=0;
            digitos<=digitos +1;
        end if;
        if digitos > 7 then
            digitos<=0;
        end if;
    end if;
```

```
case digitos is
        when 0=>
            segmentos<="00110001"; --P
            digsel <="11111110";
        when 1=>
            segmentos<="10011111"; --1
            digsel <="11111101";
        when 2=>
            segmentos<="11101111"; --_
            digsel <="11111011";
         when 3=>
            segmentos<="10011111"; --1
            digsel <="11110111";
         when 4=>
            segmentos<="01100001"; --E
            digsel <="11101111";
        when others=>
            segmentos<="11111111";--no muestra nada en display
            digsel<="11111111";
    end case;
```

Usamos una frecuencia de 1.6 ms en total para el display; es decir cada segmento estará encendido 1/8 de ese tiempo (imperceptible para el ojo) y a través de un case se va alternando entre los dígitos (de 0 a 7).

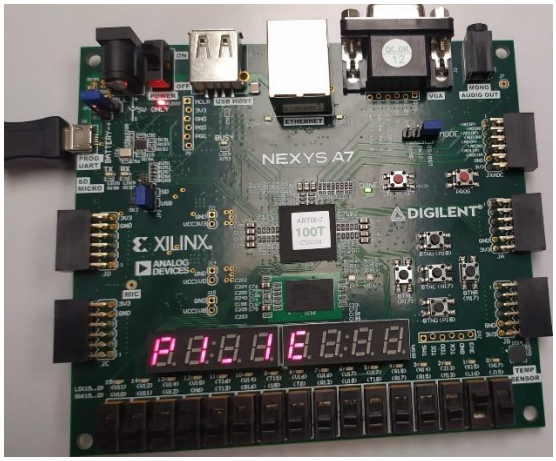
El display tiene tres “fases” de control:

No selección: esta opción se mostraría durante el reposo cuando no hay ningún producto seleccionado. El display mostraría: nO_SEL

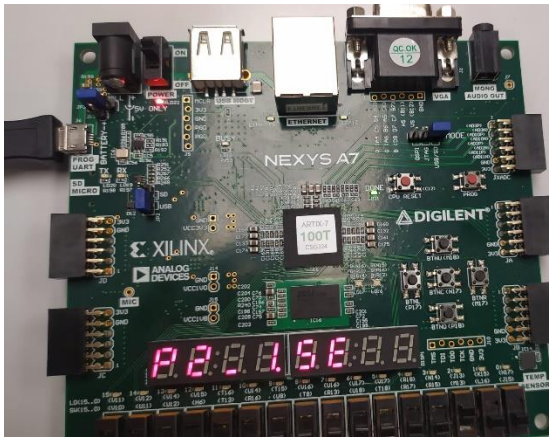


Selección de producto: a través de los switches activamos las distintas opciones que muestra el display que son el producto y su respectivo precio:

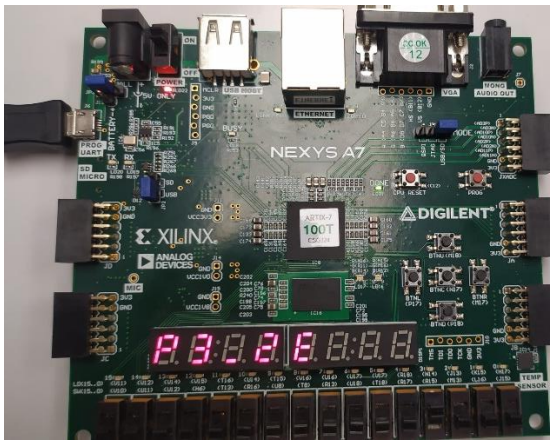
- P1_1E



- P2_1.5E

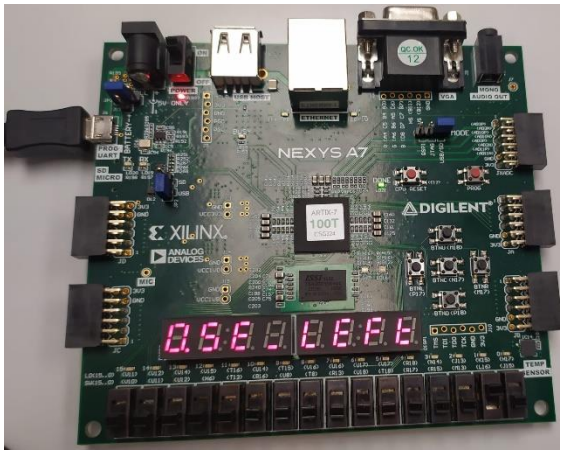


- P3_2E

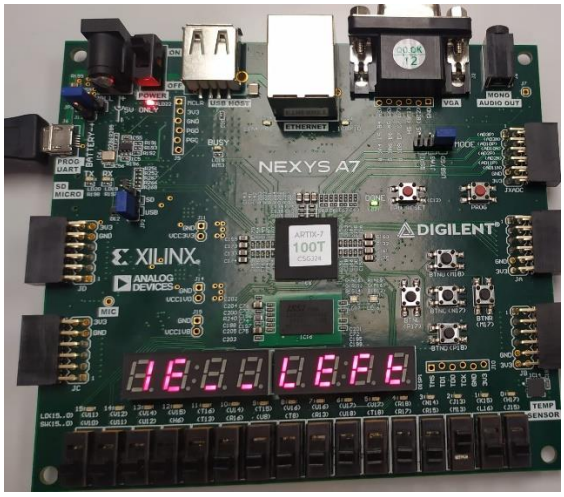


FSM (Si el producto está vendido el cambio y si no el dinero que falta) y LEDs: Para esta “fase” del display usamos dos entradas recibidas de las salidas de las FSM que son por un lado el dinero que falta:

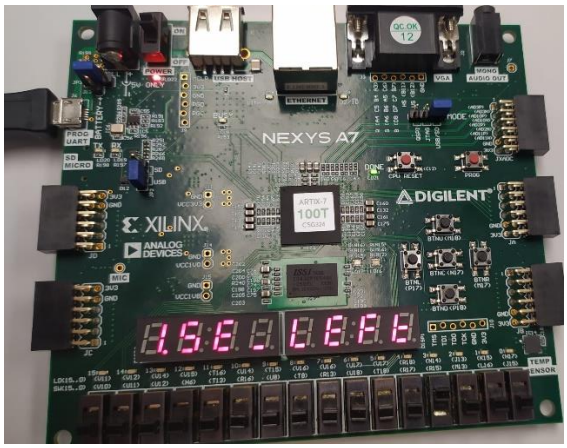
- 0.5E LEFT



- 1E LEFT



- 1.5E LEFT

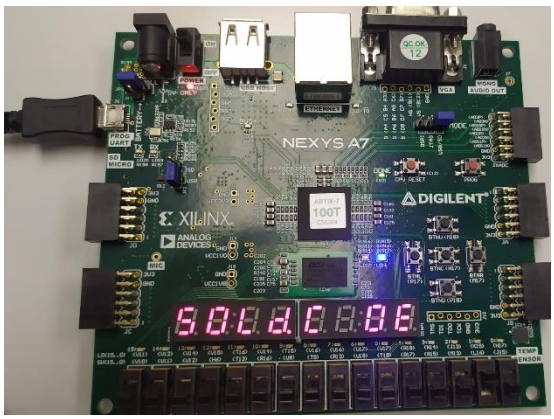


Y por otro el cambio de las FSM una vez se haya vendido el producto, es decir; si la salida de las FSM darp1 (dar producto 1) or darp2 or darp3 ‘1’ se mostrara en el display la venta del producto el cambio que se da y se encenderá un LED a saber rojo, verde o azul según el producto entonces el display mostrará las siguientes opciones.

<u>Venta del producto 1</u>: Se enciende LED AZUL:

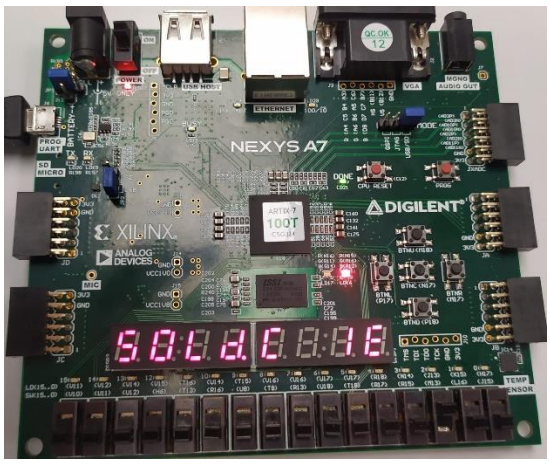
- SOLD. C0E

- SOLD.C0.5E
- SOLD.C 1E
- SOLD.C1.5E



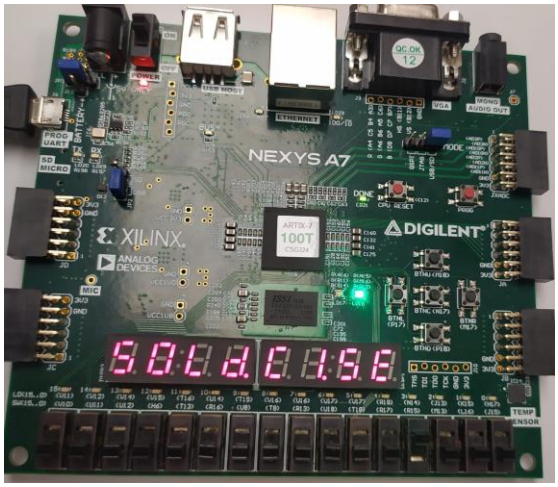
Venta del producto 2: Se enciende LED ROJO:

- SOLD. C0E
- SOLD.C0.5E
- SOLD.C 1E
- SOLD.C1.5E



Venta del producto 3: Se enciende LED VERDE:

- SOLD. C0E
- SOLD.C0.5E
- SOLD.C 1E
- SOLD.C1.5E



Sel_producto

La selección de producto se realiza a través de los switches (sw1->prod1, sw2->prod2, sw3- prod3) esta variable actúa como un enable fundamental para todo el trabajo ya que es la entrada según la cual activamos la FSM de producto correspondiente. La programación nos permite elegir indistintamente siempre y cuando las FSM se encuentren en el estado de reposo es decir podremos elegir entre los productos e ir viendo que precio tiene cada uno de elos, pero una vez se haya seleccionado un producto y se introduzca una moneda ya no se puede retroceder hasta la venta del producto o que se produzca un reset.

Producto 1

El funcionamiento de estas FSM es prácticamente el mismo en los tres productos, solo que varían los estados de cada uno.

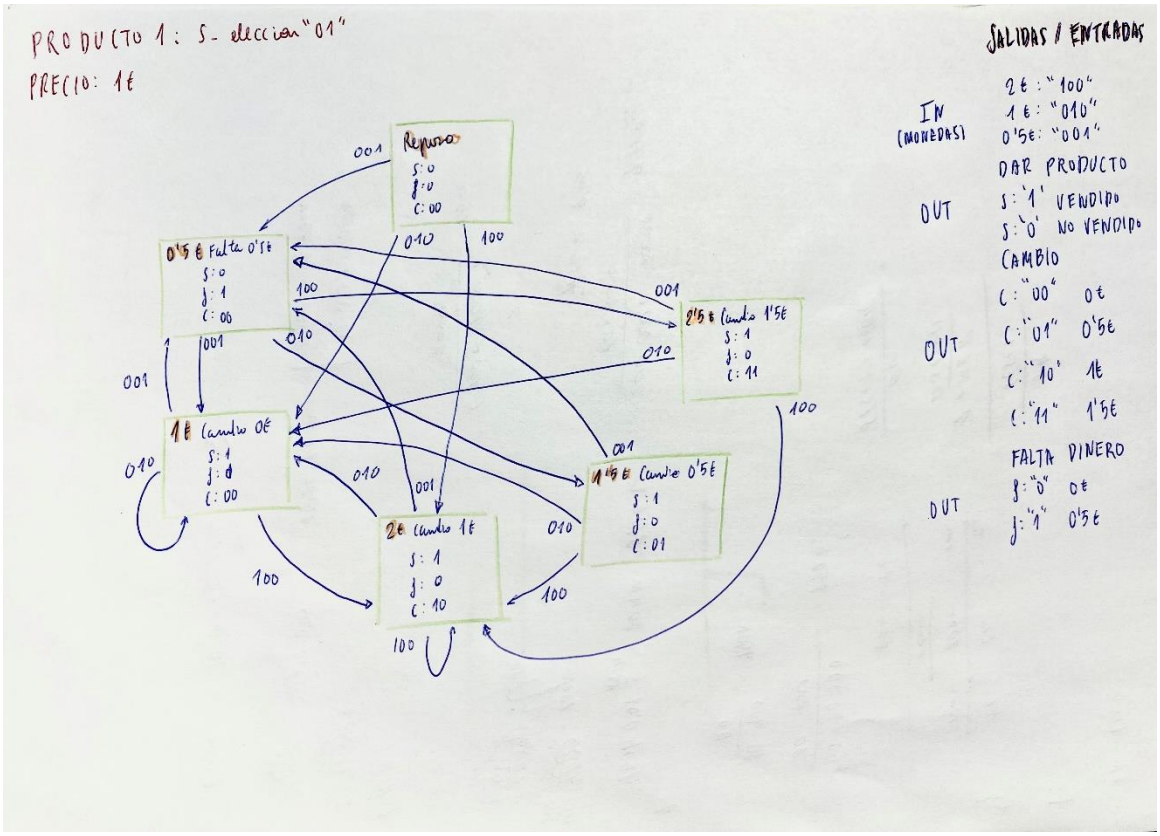
Se tratan de máquinas de moore donde cada estado contempla su propia salida. Además, las FSM solo se activa si la entrada de s_eleccion, que viene de la entidad sel_producto, es la indicada, en este caso para el producto 1 sería “01”. Esto lo hemos hecho mediante un if que englobe todo el proceso.

El producto 1 tiene un valor de 1euro y se trata de una máquina de moore de 6 estados, donde cada uno define el valor de sus salidas. Estas salidas son, darp1 (si has introducido todo el dinero te da el producto), falta1 (dinero que falta hasta llegar al precio del producto), y cambio1(que en casa de darte el producto si te ha sobrado dinero te lo da).

La salida darp1 es un valor de un bit, ya que '1' es dar producto, y '0' no dar producto. Falta1 es de un bit, '0' no falta nada, '1' falta 50cent. Cambio1 es también de dos bits, “00” no hay cambio”, “01” 50cent de cambio, “10” 1 euro de cambio y “11” 1,5euro de cambio.

Las entradas que hacen posible cambiar de estados son tres, dependiendo del dinero que tengas estarás en un estado o en otro, estas entradas son, “001” 50cent, “010” 1euro, “100” 2euros.

Como se ha comentado antes, dependiendo del estado las salidas tienen un valor u otro, pero los estados dependen del dinero que hayas metido, por ejemplo, si estas en el estado inicial que no hay dinero y metes 1euro (“010”) pasarás al estado 1euro, donde las salidas son darp1='1', falta1='0', y cambio1=”00”.



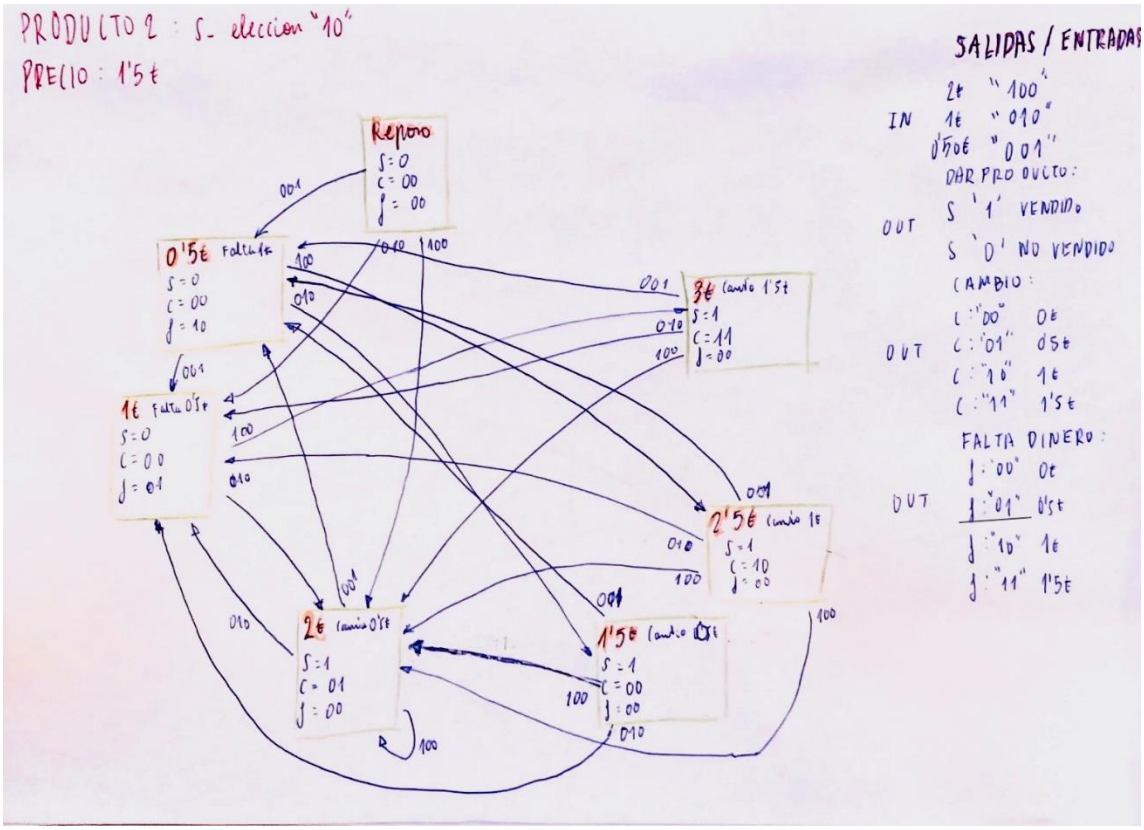
Producto 2

La FSM del producto 2 cuesta 1.5euros, solo estará activa cuando la elección sea la correspondiente a este producto “10”, con una entrada de 0 en el reset la maquina se reiniciaría.

Esta máquina consta de 7 estados en los cuales tenemos tres salidas, cambio (indica el cambio que da la maquina tras la venta de un producto), falta (el dinero que falta por introducir para realizar la compra del producto) y dar (varía entre 0 y 1 para indicar la venta del producto). En este caso la salida falta2 es de dos bits, “00” no falta nada, “01” falta 50cent, “10” falta 1 euro. Las variables cambio2 y darp2 son iguales que en el producto 1.

Proceso de estados:

Inicialmente la maquina recibe como condición necesaria el enable de producto que indica la selección de este, luego partiendo siempre del estado de reposo recibe la entrada de dinero (modificado convenientemente a un vector de 3 bits a saber 001 “50 cents”, 010 “1 euro” y 100 “2 euros”). Tras recibir la entrada la maquina evoluciona teniendo en cuenta en todos los casos el cambio y el dinero que falta para dar el producto. Cuando se produce la venta darp2 = ‘1’, en caso de desear otro producto 2 simplemente tendríamos que volver a introducir el importe de nuevo, pero si quisiéramos otro debemos pulsar el botón de inicio de compra que nos permite seleccionar de nuevo entre los productos disponibles.



Producto 3

El funcionamiento de esta FSM del producto 3 tiene el mismo funcionamiento que las otras dos FSM.

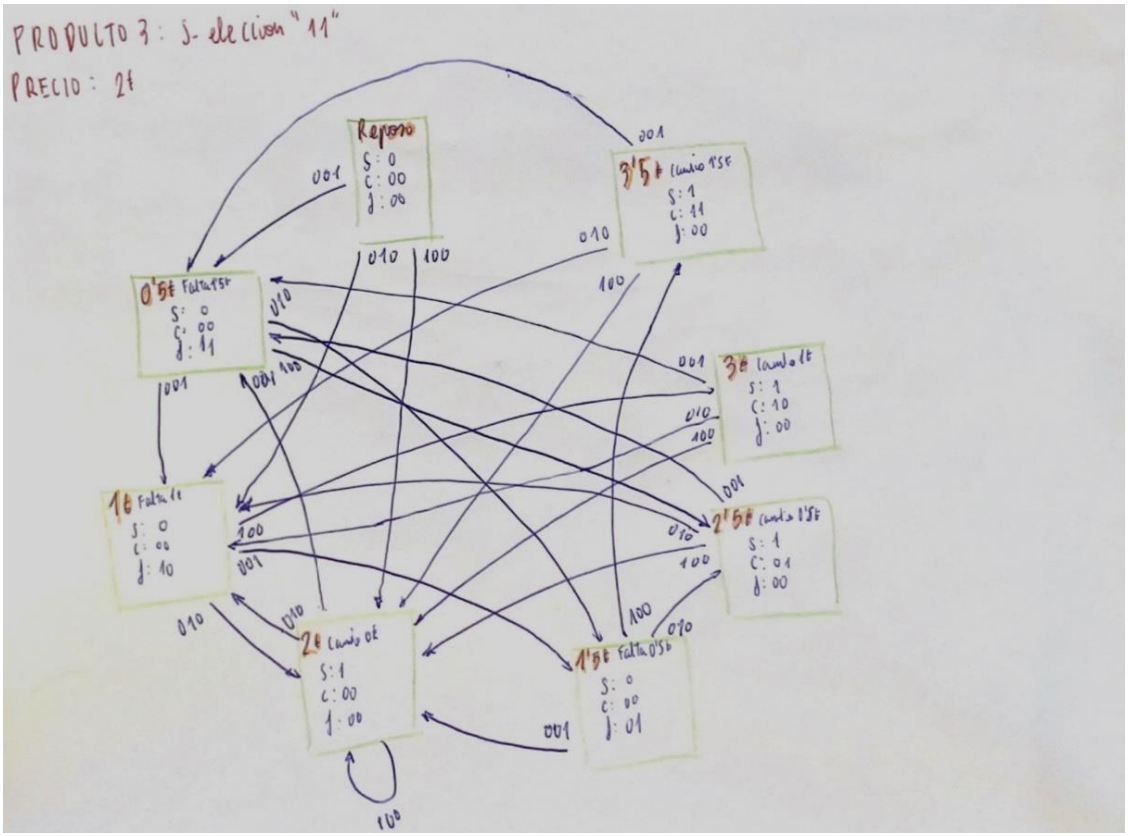
En este caso se trata de un producto cuyo precio son 2euros, y la máquina de estados está compuesta por 8 estados. La entrada de s_eleccion (selección del producto desde los switches) que tiene que estar habilitada en este caso sería la ”11”.

Tenemos las mismas salidas que en las otras FSM, es decir, darp3 (dar producto 3), falta3 (dinero necesario para llegar al precio del producto, y cambio3 (en caso de haber metido todo el dinero y sea necesario el cambio). La máquina de estados está pensada para que las salidas dependan de los propios estados, por lo que solo hemos tenido que declarar en cada estado las salidas correspondientes. Los bits de todas las salidas son iguales que en el producto 2, pero en este caso en la variable falta3 también se usaría el “11” que sería falta 1.5euros.

Las entradas vuelven a ser las mismas, “001” 50cent, “010” 1euro, y “100” 2euros. Esta FSM tiene un estado más que la anterior debido a que es posible estar en el estado donde tenemos 1,5euros y metamos 2euros, que nos iríamos a un estado de 3,5euros, que en el producto anterior sería imposible porque al tener 1,5euros ya nos daría el producto y 0euros de cambio.

También esta implementado el reset, y el botón de inicio de compra de otro producto en caso de querer cambiar de producto, si queremos otro producto igual solo hay que seguir metiendo monedas.

En este caso si la máquina nos da el producto se encenderá el LED verde.



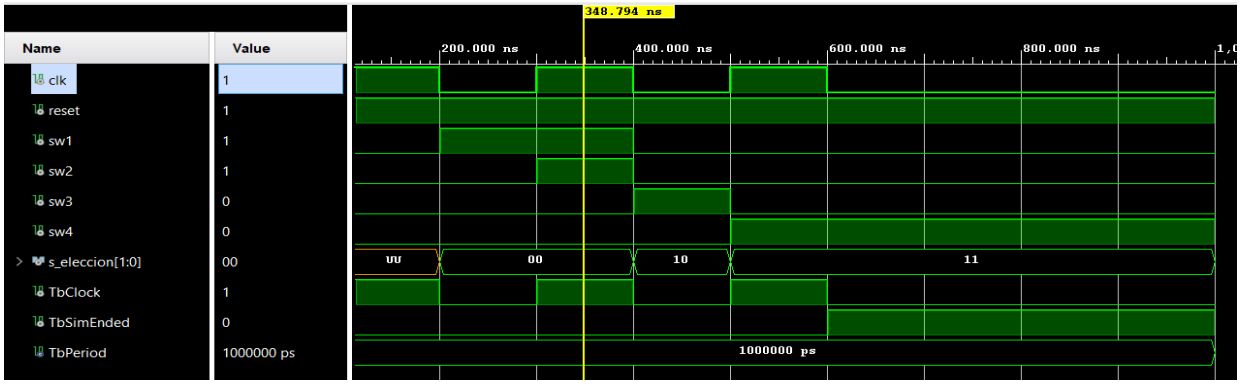
PRUEBAS Y SIMULACIONES

Test benches

Tras la gestación de un modelo que cumpliese las expectativas de diseño, se ha procedido a realizar numerosas pruebas mediante simulaciones. Estas simulaciones han servido para llevar un orden del progreso del proyecto, ya que ofrecen una información detallada del funcionamiento de diversos sistemas de manera individual.

Como se ha comentado, las simulaciones se han realizado de manera específica para cada entidad/máquina de estado, lo cual permite el análisis del sistema dividido. Ha de recalcarse que los valores que se han dado a las entradas de cada entidad de simulación han sido arbitrarios y variados para la visualización de salidas de la misma índole.

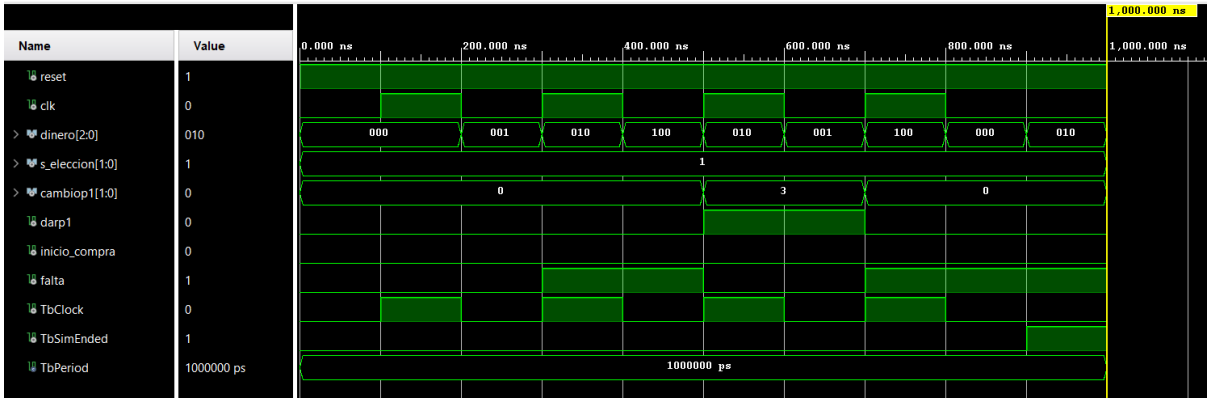
Para empezar, se ha realizado un testbench del selector de producto, una entidad simple pero crucial para el correcto funcionamiento del sistema, ya que es la primera parte por la que pasa el ciclo de la máquina y la que da salida a la compra de los productos.



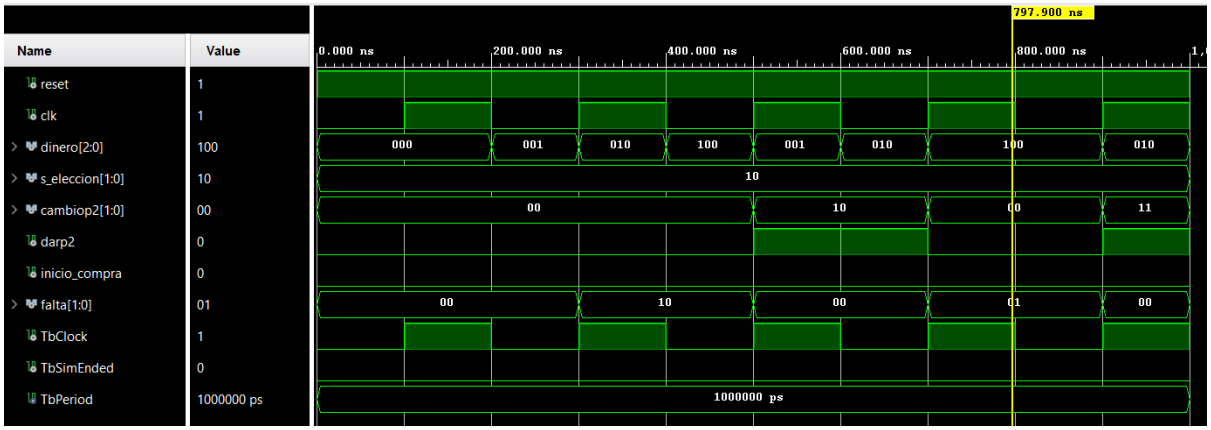
Testbench Selector de producto

Como se puede observar, los resultados son variados, pues en la simulación se han seleccionado todos los productos al menos una vez. Puede notarse que, al comienzo de la simulación, la salida no ha adquirido ningún valor concreto, pues la entrada es nula (no selección).

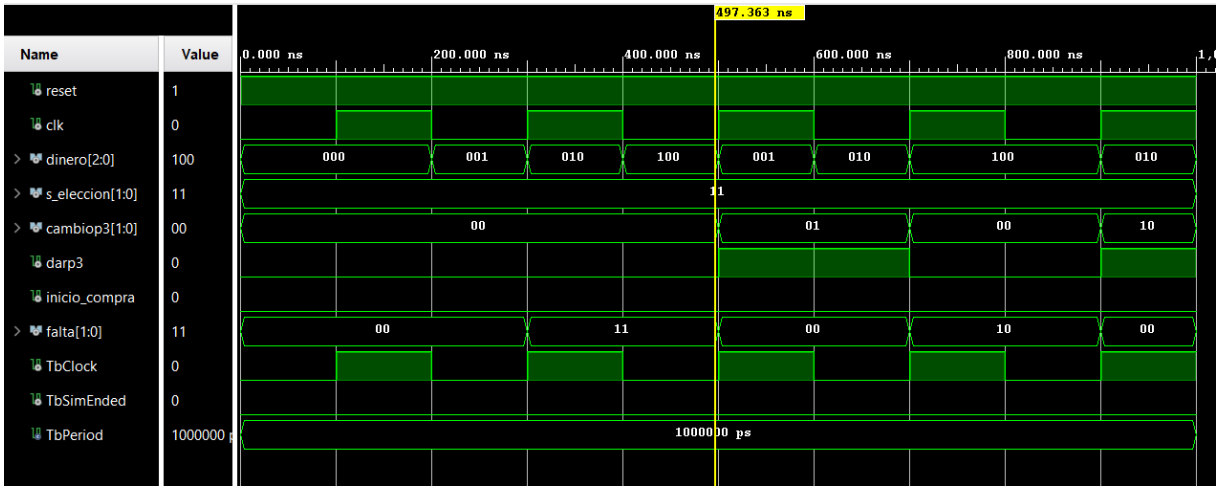
Las siguientes simulaciones que se han realizado han sido las de las máquinas de estado o FSM que gobiernan el ciclo de compra/venta de los productos. En ellas, la única entrada variable es el dinero que se introduce por medio de los pulsadores de la placa Nexys4DDR, y que se ha representado como un vector de tres bits (cada bit es una moneda diferente).



Testbench Producto 1



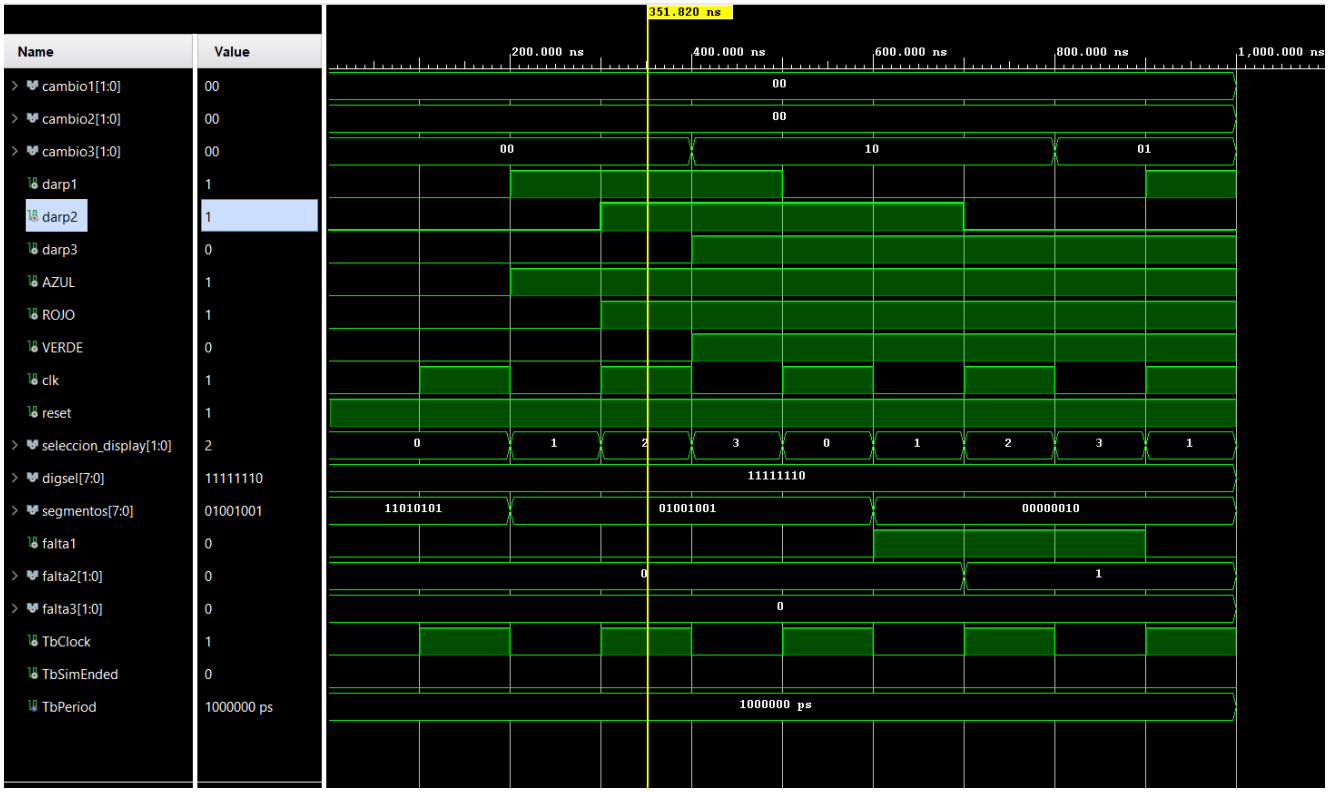
Testbench Producto 2



Testbench Producto 3

Se puede observar que, en la primera máquina, Producto 1, la salida “falta” que representa un aviso al usuario para comunicar que el dinero introducido no es suficiente para la compra del producto, es un bit, mientras que en las dos máquinas restantes es un vector de dos bits. Esto es debido a la propia naturaleza de la máquina, pues en el Producto 1 el precio es de 1 € y los estados intermedios que pueden darse al introducir el dinero son menores.

Ya para finalizar, se ha realizado una simulación de la máquina de estados que gobierna las salidas del sistema, displays de 7 segmentos y leds. En esta entidad, las entradas son las salidas de las entidades anteriores.



Algunos resultados se han representado en decimal en vez de binario en el diagrama, pero el valor sigue siendo el mismo.

Pruebas Hardware

Una vez se ha llegado a una versión del modelo completa y homogénea, se ha procedido a la carga del programa principal en la tarjeta Nexys 4DDR. Las pruebas realizadas con el hardware han sido arduas y tediosas, ya que la herramienta es lenta a la hora de la subida del programa a la placa, lo cual ha supuesto un periodo de tiempo para tener en cuenta. Se han ido realizando mejoras en el programa, ya que la realidad del funcionamiento de este dispositivo es muy dispar de las simulaciones mediante test bench. Una vez el funcionamiento se ha adecuado a los requerimientos de un usuario normal, el programa está listo para mostrarse y evaluarse.

FUNCIONAMIENTO DEL PROGRAMA

https://drive.google.com/file/d/19jVUw5stBcEssjfq19WBLEoWGDzkZu7r/view?usp=share_link

REPOSITORIO ONLINE

Para el desarrollo de este trabajo se ha utilizado la herramienta online Github para llevar un seguimiento de las tareas realizadas. Se adjunta un enlace al repositorio.

https://github.com/ismael45/Maquina-Expendedora

CONCLUSIONES Y POSIBLES MEJORAS

En conclusión, en este trabajo se ha implementado una máquina expendedora en VHDL utilizando la placa Nexys 4DDR y se ha verificado su funcionamiento a través de pruebas de simulación y de laboratorio. Se ha demostrado que la placa Nexys 4DDR es una plataforma adecuada para el desarrollo de sistemas digitales basados en VHDL, ya que cuenta con un procesador ARTIX-7 y una amplia variedad de periféricos y componentes digitales. Además, se ha utilizado la herramienta Vivado para la implementación y simulación del diseño, lo que ha facilitado el proceso de desarrollo y ha permitido obtener resultados precisos y fiables. En resumen, este trabajo demuestra la capacidad de VHDL para describir y controlar sistemas digitales complejos y su aplicabilidad en el diseño de dispositivos embebidos utilizando la placa Nexys 4DDR.

La funcionalidad de nuestra máquina expendedora es muy completa. Una vez terminado el proyecto planteamos la idea de poder cambiar de producto en el caso de arrepentirte de tu elección mientras estas metiendo monedas para el producto elegido, y que te devolviera el dinero metido hasta el momento para poder usarlo para otro producto o que se quedara guardado para usarlo en el producto elegido. En nuestro caso, podrías cambiar de producto usando el botón de iniciar compra de otro producto, pero el dinero introducido hasta el momento se lo tragaría la máquina. Pensamos que se podría introducir en nuestro programa creando una variable dentro de cada fsm de los productos, que se encargara de contar el dinero hasta el momento y tuviera memoria para poder usarlo en caso de elegir otro producto.