TRABAJO SED VHDL

GRUPO Nº. 15

MÁQUINA EXPENDEDORA

INTEGRANTES

FRANCISCO MARTÍN CUSCURITA 55972

JIMENA LÓPEZ MALDONADO 55951

JOAN BELLIDO INES

ÍNDICE

1.- INTRODUCCIÓN Y OBJETIVOS

1.- INTRODUCCIÓN Y OBJETIVOS

1.1.- Enunciado

En esta sección del proyecto de la asignatura se llevará a cabo la implementación de un diseño en la placa de laboratorio NEXYS 4 DDR que resuelva uno de los enunciados propuestos en Moodle para los estudiantes. En nuestro caso, se ha elegido el problema de la máquina expendedora. El enunciado de la propuesta de diseño es el siguiente:

*“Diseñe una máquina expendedora de refrescos. Admite monedas de 10c, 20c, 50c y 1€. Sólo admite el importe exacto, de forma que si introducimos dinero de más da un error y “devuelve” todo el dinero. Cuando se llega al importe exacto del refresco (1€) se activará una señal para dar el producto. Como entradas tendrá señales indicadoras de la moneda, señales indicadoras de producto y como salidas la señal de error y la de producto”.*

1.2.- Objetivos

El principal propósito del trabajo es el diseño de la máquina expendedora como se propone en el enunciado. Además de esto, se han pensado más funcionalidades que se implementarán en el diseño inicial, como son:

* Añadir un refresco adicional a la máquina expendedora.
* Personalizar el precio de los refrescos sin realizar grandes modificaciones en el código.
* Utilizar el display de 7 segmentos visto en las prácticas de laboratorio para:
* Botón de reset con el que se vuelva al estado inicial.

2.- DESCRIPCIÓN DEL PROYECTO

2.1.- Funcionamiento

Para el diseño y estructura del proyecto, se utilizarán como referencia los códigos de las prácticas de la asignatura, que servirán de base para desarrollar el resto del código. Por lo tanto, los aspectos relacionados con el diseño de máquinas de estado, la sincronización, la detección de flancos y el manejo de los displays seguirán una estructura similar a la presentada en estas prácticas.

El funcionamiento de la máquina expendedora se describe de la siguiente manera: se comienza en un estado inicial en el que se debe seleccionar el tipo de refresco, ya sea el tipo 1 o el tipo 2. Antes de introducir monedas, se debe confirmar la intención de proceder con el pago. A continuación, se introducen las monedas mientras el display indica la cantidad restante por depositar. Esta funcionalidad está a cargo tanto del contador como del controlador del display.

Si se introduce la cantidad exacta de dinero, aparecerá un mensaje indicando que el refresco seleccionado ha sido dispensado. Si la cantidad es incorrecta, se mostrará un mensaje de error y las monedas serán devueltas. Finalmente, se regresa al estado inicial al presionar el botón de reinicio o, si la operación se ha completado correctamente, al establecer el valor de pagar en ‘0’ nuevamente.

2.2.- Entradas y salidas

Entradas

1. **CLK (Reloj)**
   * Señal de reloj principal de la placa con una frecuencia de **100 MHz**.
   * Todas las señales del sistema, **excepto la señal RESET**, están sincronizadas con este reloj.
2. **RESET (Reinicio)**
   * Señal de reinicio del sistema que **retorna la máquina a su estado inicial**.
   * Es una señal asíncrona respecto al reloj principal.
3. **MONEDAS (Vector de Entradas)**
   * Vector de **4 bits** que indica la moneda introducida en la máquina. Cada combinación corresponde a un valor específico:
     + **“0001”**: Moneda de **10 céntimos (10c)**.
     + **“0010”**: Moneda de **20 céntimos (20c)**.
     + **“0100”**: Moneda de **50 céntimos (50c)**.
     + **“1000”**: Moneda de **1 euro (1€)**.
4. **PAGAR (Confirmación de Pago)**
   * Señal de control que **confirma la selección del producto** y da inicio al proceso de pago.
   * Una vez activada, se verifica si el importe introducido es suficiente y se procede a la entrega del refresco.
5. **TIPO\_REFRESCO (Selección del Producto)**
   * Vector de **2 bits** que permite **elegir el refresco** deseado. Las opciones disponibles son:
     + **“01”**: Selección del **Refresco 1**.
     + **“10”**: Selección del **Refresco 2**.
   * Cualquier otra combinación de bits se considera no válida.

Salidas

1. **ERROR (Indicador de Error)**
   * Señal conectada a **un LED de la placa**.
   * Se activa si el importe introducido **excede el precio del refresco seleccionado**, indicando un error en la operación.
2. **REFRESCO\_OUT (Entrega del Producto)**
   * Señal conectada a **otro LED** que confirma la **entrega del refresco seleccionado**.
   * Esta salida se activa cuando se ha introducido el importe exacto para el refresco.
3. **ESTADOS (Vector de Estados)**
   * Vector de señales que representa **el estado actual de la máquina**.
   * Su función es puramente **informativa** y permite el seguimiento del flujo de ejecución del sistema.
4. **SEGMENTOS y DIGCTRL (Control de Displays de 7 Segmentos)**
   * **SEGMENTOS**: Salida que contiene la **información del número o letra** que se mostrará en los displays de 7 segmentos.
   * **DIGCTRL**: Señal de control que **activa el display específico** que debe mostrar la información proporcionada.
   * Ambas señales operan a una frecuencia de reloj **reducida** respecto al CLK principal, gracias al **prescaler** incorporado en el diseño.

Notas Adicionales

* El sistema está diseñado para operar de manera sincronizada, excepto por el RESET, que funciona de manera asíncrona.
* El control de pagos y selección de productos se basa en la correcta combinación de las señales MONEDAS, PAGAR y TIPO\_REFRESCO.
* Las salidas proporcionan retroalimentación visual a través de LEDs y displays de 7 segmentos, facilitando la interacción con el usuario.

2.3.- Diagrama de estados

Imagen de la pantalla de un celular

Descripción generada automáticamente con confianza media

La máquina de estados tiene cuatro estados principales: Reposo, Etapa de Pago, Etapa de Error y Etapa de Salida de Refresco.

Primero, en el **estado de Reposo**, la máquina está inactiva, pero en el **display** se muestra cuál es el refresco que está seleccionado. Este es el estado inicial y también es al que siempre regresaremos si se pulsa el botón **RESET**, sin importar en qué etapa nos encontremos.

Después, al entrar en la **Etapa de Pago**, el usuario tiene que introducir las monedas para pagar el refresco. Aquí, un **contador** va sumando el importe que se va introduciendo y calcula si es correcto o no. Dependiendo de eso, el contador manda una señal:

* **PAGO\_OK** si el importe introducido es exacto.
* **ERROR\_COUNTER** si se pasa del importe permitido o hay algún error.

La señal que se envía determinará el **próximo estado**:

* Si la señal es **PAGO\_OK**, pasaremos a la **Etapa de Salida de Refresco**.
* Si la señal es **ERROR\_COUNTER**, pasaremos a la **Etapa de Error**.

En la **Etapa de Error**, el **display** avisará al usuario de que ha habido un error, mientras que en la **Etapa de Salida de Refresco**, el **display** mostrará que el refresco ha sido entregado correctamente.

Por último, siempre existe la opción de volver al **estado de Reposo** de dos maneras:

1. Automáticamente, dependiendo del estado en el que nos encontremos.
2. Pulsando el botón **RESET**, que siempre devuelve la máquina al estado inicial.

Además, en cada estado, el **display** proporciona información importante:

* En **Reposo**, muestra el refresco seleccionado.
* En **Pago**, indica el precio total y la cantidad de dinero que falta por introducir.
* En **Salida de Refresco**, informa que el refresco ha salido.
* En **Error**, avisa de que ha ocurrido un problema.

3.- DESARROLLO DE ENTIDADES Y SIMULACIÓN.

3.1.- Esquema de identidades

Una vez establecidos los objetivos y especificaciones que debe cumplir el proyecto, procederemos a planificar la estructura que adoptará nuestro programa para llevar a cabo adecuadamente cada una de sus funcionalidades.

El diseño del programa se desarrollará de manera que incorpore diversas entidades que interactuarán de forma coordinada, intercambiando datos a través de variables auxiliares que enlazarán sus entradas y salidas. Asimismo, cada entidad será responsable de ejecutar un conjunto de tareas distintivas que detallaremos a continuación.

En primer lugar, encontramos las entidades **SYNCHRNZR** y **EDGE\_DETECTOR**, que, tal como hemos observado en las prácticas, se ocupan respectivamente de sincronizar las entradas con el reloj y de identificar los cambios de estado o flancos. Su utilización resulta fundamental para minimizar posibles errores durante el proceso de lectura.

Por otro lado, contamos con tres entidades principales que estructuran el programa: **COUNTER**, **DISPLAY\_CONTROL** y **FSM**. La primera de ellas se encarga de contabilizar las monedas introducidas y de verificar que el importe depositado coincide con el solicitado. Mediante la entidad **DISPLAY\_CONTROL** gestionaremos la visualización en los displays de la placa, ajustándola dinámicamente conforme el usuario avanza en la selección y el pago del producto. Finalmente, la entidad **FSM** actúa como el núcleo de coordinación del sistema, dirigiendo el funcionamiento de la máquina basándose en el diagrama de estados previamente diseñado.

El **DISPLAY\_CONTROL** está vinculado a otra entidad, denominada **DECODE**, cuya función es activar los segmentos necesarios para representar, en los displays, los números o letras transmitidos desde la unidad de control.

Además, disponemos de una entidad superior llamada **MAQ\_EXP**, que no aparece reflejada en el diagrama, pero cuya función principal es invocar a las distintas entidades con sus correspondientes parámetros y definir las variables auxiliares necesarias para su interconexión.

Una vez comprendido el rol de cada una de las entidades, pasaremos a analizar el flujo de ejecución del programa, utilizando como referencia tanto el diagrama de estados como el de entidades.

El sistema de la máquina expendedora comienza en un estado de reposo. Se activa al detectar, por medio de la **FSM**, que la entrada de **PAGAR** ha sido activada y se ha seleccionado una de las dos opciones de refresco disponibles. Esto nos lleva al **COUNTER**, que registrará las monedas insertadas. Simultáneamente, gracias a **DISPLAY\_CONTROL**, los displays mostrarán el producto seleccionado, su precio y la cantidad restante para completar el pago. El contador comunicará su estado a la **FSM** mediante las señales **PAGO\_OK** y **ERROR**, proporcionando a esta última la información necesaria para determinar si el producto está listo para ser dispensado.

En la siguiente página se puede observar el diagrama de entidades.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

3.2.- SYNCHRONIZER

Forma, Cuadrado

Descripción generada automáticamente

Cuando las señales asíncronas se emplean directamente en la lógica sincronizada con el reloj del sistema, pueden producirse fallos en la lógica digital, lo que resalta la relevancia de esta entidad.

El **SYNCHNZR** desempeña un papel crucial para garantizar el funcionamiento fiable y consistente de la máquina expendedora, proporcionando al usuario una experiencia segura y uniforme. Su tarea principal consiste en recibir señales externas que no están sincronizadas con el reloj del sistema y alinearlas correctamente. Este proceso minimiza el riesgo de condiciones metaestables, en las que una señal podría quedar en un estado indefinido entre '0' y '1', lo que podría ocasionar un comportamiento impredecible del sistema.

En la definición de esta entidad se incluyen dos parámetros genéricos: **Num\_Tipo\_Refrescos** y **Num\_Monedas**, que corresponden a la cantidad de tipos de refrescos ofrecidos y a las distintas denominaciones de monedas aceptadas, como se explicó anteriormente.

Las entradas de la entidad reflejan las decisiones del usuario: las monedas introducidas están representadas por el vector **AS\_MONEDAS[3 TO 0]**; el bit **AS\_PAGAR** adopta un nivel alto cuando el usuario presiona el botón de pagar, y la selección del tipo de refresco se transmite mediante la señal **AS\_TIPO\_REFRESCO**.

En la arquitectura de la entidad, se declaran tres señales que, en un primer proceso, capturan los valores de sus respectivas entradas en cada flanco positivo del reloj. Posteriormente, un segundo proceso transfiere estas señales a un segundo registro en el siguiente borde ascendente del reloj. Este enfoque contribuye a reducir aún más el riesgo de condiciones metaestables, ya que permite que cualquier señal que haya quedado en un estado indefinido durante la primera captura sea estabilizada antes de llegar a la lógica principal del sistema.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Una captura de pantalla de una computadora

Descripción generada automáticamente