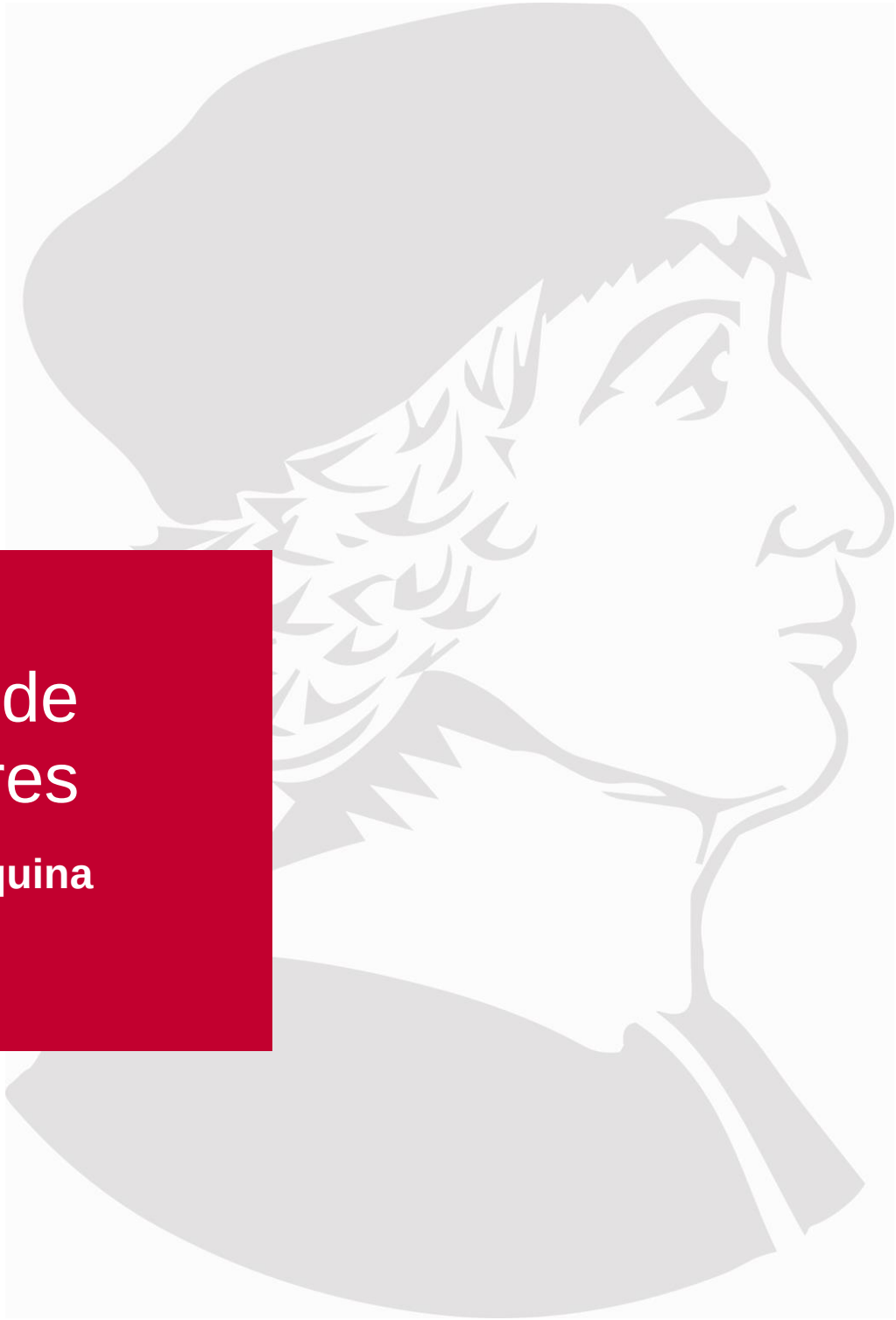
UNIVERSIDAD NEBRIJA

Álvaro Pereda Sánchez
Santiago Valderrama Flores
Pedro Zuñeda Diego



Tecnología de computadores

Trabajo Final: Máquina Expendedora

UNIVERSIDAD NEBRIJA

- **Índice** **2**

UNIVERSIDAD NEBRIJA

Introducción al proyecto

En el marco de este proyecto, se aborda el desafío de diseñar el sistema de control para una máquina expendedora de bebidas que ofrece latas de refrescos a un precio fijo de 2€ cada una. El objetivo principal es garantizar una operación eficiente y segura, permitiendo a los usuarios realizar transacciones mediante la inserción de monedas de 1€ y 2€, así como billetes de 5€.

Para lograr este propósito, se propone la implementación de un sistema basado en una máquina de estados. Este enfoque proporciona una estructura lógica y ordenada que facilita la comprensión y el diseño paso a paso del comportamiento deseado de la máquina.

A lo largo del proyecto, se consideran aspectos cruciales para la experiencia del usuario, como la devolución de cambio en caso de que se introduzca una cantidad mayor de dinero que el costo de la lata. Además, se establece que la máquina siempre dispone de suficientes latas para la venta.

1. Descripción del trabajo

Para la creación de la máquina dispensadora, hemos creado un código de VHDL que consta de dos estados: el estado inicial, denominado "*INIT*", y un estado auxiliar llamado "*S1*". Esta máquina de estados sigue un modelo Mealy, ya que las salidas no dependen del estado actual, lo cual se ha elegido con el propósito de simplificar y mejorar la visualización de la máquina. Además, se ha incorporado un reset asíncrono activo a nivel bajo para asegurar el correcto funcionamiento de la máquina de estados.

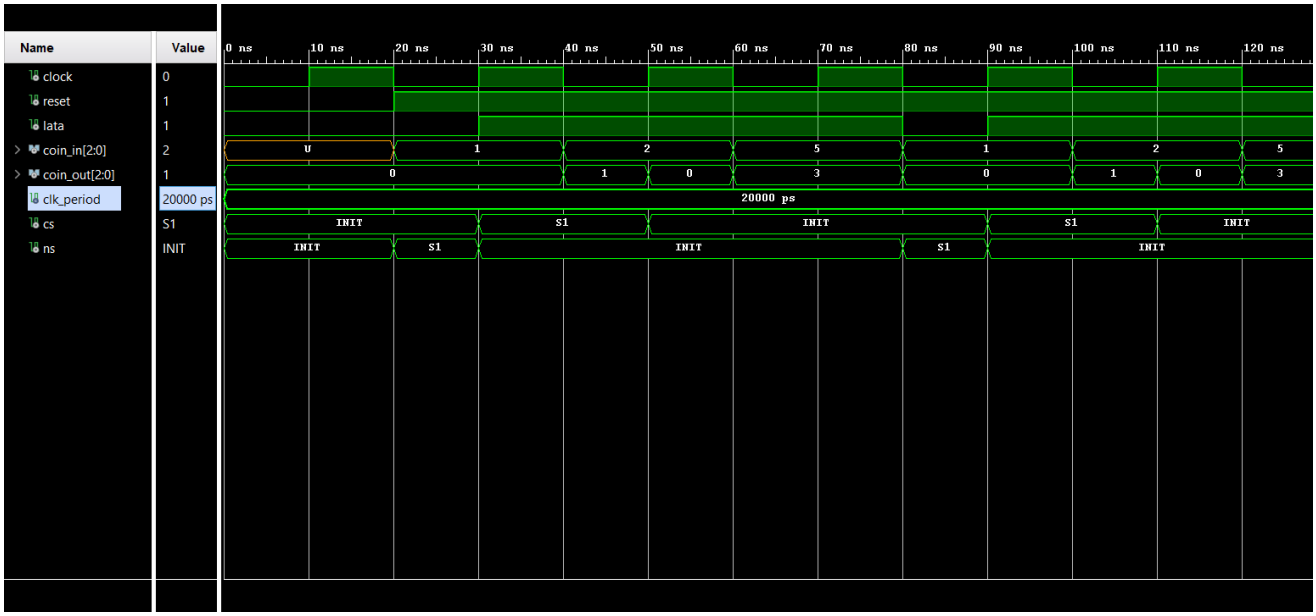
2. Explicación de los códigos VHDL. Simulación del trabajo

Para la primera máquina de estados hemos declarado las bibliotecas necesarias y una entidad que cuenta con tres entradas: “*reset*”, “*clock*”, y “*coin_in*”. Esta última es un vector de 3 bits, elegido para representar todas las posibles monedas que el usuario pueda introducir en la máquina expendedora. Además, la entidad cuenta con dos salidas: “*lata*”, activa a nivel alto cuando se detecta que el usuario ha introducido la cantidad necesaria para comprar una lata, y “*coin_out*”, un vector de 3 bits que representa la cantidad que la máquina debe devolver en caso de que el usuario introduzca una cantidad superior al precio de la lata.

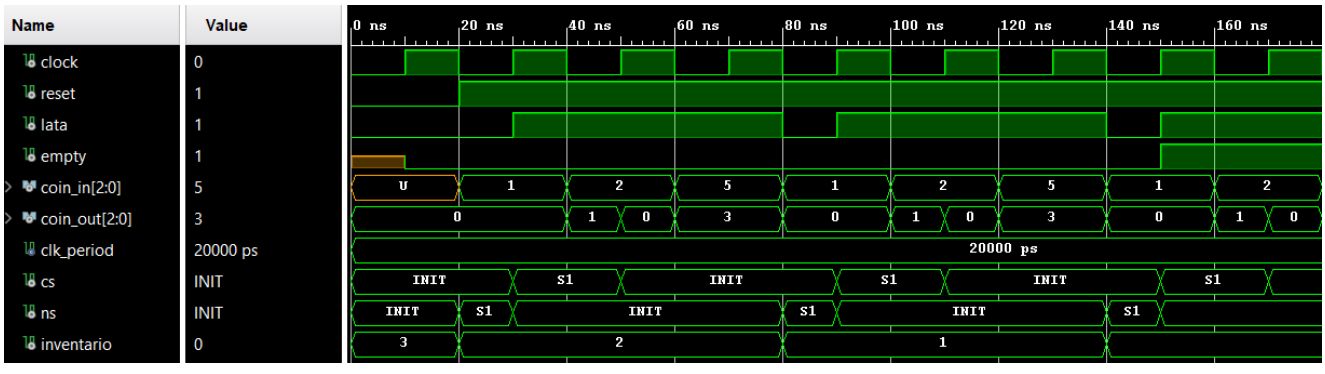
En primer lugar, declaramos un tipo enumerado state_type con dos posibles estados: INIT y S1. También, se definen las señales cs y ns, que representan el estado actual y el próximo estado.

El código se estructura en dos procesos. El primero maneja la lógica de cambio de estado en función de la señal de reloj asíncrona, con una lista de sensibilidad de clock y reset. Si la señal de reinicio es baja, el estado actual (cs) se establece en INIT. En el flanco de subida del reloj, el estado actual se actualiza al próximo estado (ns).

El segundo proceso implementa la lógica de la máquina de estados Mealy, utilizando la lista de sensibilidad. Dependiendo del estado actual (cs) y la entrada de monedas (coin_in), se determina el próximo estado (ns) y las salidas (lata y coin_out).



Este es el diagrama de tiempos de la máquina dedicada a la parte de ampliación.



La entidad cuenta con tres entradas: "*reset*", "*clock*" y "*coin_in*". La entrada "*reset*" se utiliza para reiniciar, la entrada "clock" se utiliza para controlar el paso del tiempo, y la entrada "*coin_in*" se utiliza para representar las monedas que el usuario introduce en la máquina expendedora. La entidad también cuenta con dos salidas: "lata" y "*coin_out*". La salida "lata" se activa a nivel alto cuando la máquina de estados detecta que el usuario ha introducido la cantidad necesaria para comprar una lata, y la salida "*coin_out*" representa la cantidad de monedas que la máquina debe devolver al usuario en caso de que introduzca una cantidad superior al precio de la lata.

En la arquitectura se declara un tipo enumerado "*state_type*" con dos posibles estados: "*INIT*" y "*S1*". También, se definen las señales "*cs*" y "*ns*", que representan el estado actual y el próximo estado.

El código de la arquitectura se estructura en dos procesos. El primer proceso maneja la lógica de cambio de estado de la máquina de estados. Este proceso se activa en el flanco de subida del reloj. Si la señal de reinicio es baja, el estado actual se establece en "INIT". De lo contrario, el estado actual se actualiza al próximo estado.

El segundo proceso implementa la lógica de la máquina de estados. Este proceso se activa en el flanco de subida del reloj. Dependiendo del estado actual y de la entrada de monedas, se determina el próximo estado y las salidas de la máquina de estados.

Para introducir la mecánica de rellenar el inventario de la máquina expendedora en el diseño, se podría hacer de varias maneras, pero todas requerirían de una nueva señal de entrada que indique que se ha rellenado el inventario (se supone que siempre que se rellena se deja el inventario a 3, y no a 2 o 1). Una manera adaptar el código y las sentencias condicionales para que, si en cualquiera de los dos estados se activa la señal de rellenar, la señal de *inventario* se reestablezca a 3.

UNIVERSIDAD NEBRIJA

3. Análisis de tiempos, áreas y consumos

Este reporte explica que recursos físicos se han usado del chip (LUTs, registros, multiplexores, etc). El código de la primera máquina expendedora usa un total de 4 LUTs como lógica y 1 Registro como biestable. Una LUT (**L**ook **U**p **T**able), es una tabla que guarda la información sobre el comportamiento de ciertos componentes electrónicos como puertas lógicas o biestables, aunque el término es ampliamente utilizado en muchos otros entornos como la conversión de archivos o fotografía. En este reporte se explica como se utilizan estas LUTs, los registros y los multiplexores del chip, junto con el número disponible y las utilizadas por nuestro diseño.

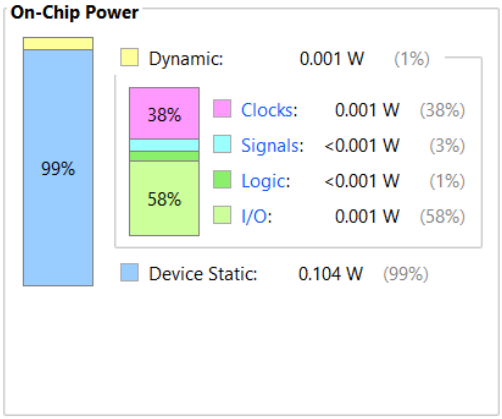
Site Type	Used	Fixed	Available	Util%
Slice LUTs	4	0	53200	<0.01
LUT as Logic	4	0	53200	<0.01
LUT as Memory	0	0	17400	0.00
Slice Registers	1	0	106400	<0.01
Register as Flip Flop	1	0	106400	<0.01
Register as Latch	0	0	106400	0.00
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

En esta imagen se puede apreciar cuanto consume el código, tanto de forma dinámica, cuando está en ejecución; como de forma estática, que es el consumo cuando simplemente se conecta a la corriente. Además, se puede observar la **Potencia Total en el Chip** (On-Chip Power), que es la suma del consumo dinámico y el consumo estático. Es evidente que nuestro diseño apenas usa potencia dinámica, y que prácticamente toda la potencia suministrada al chip es estática. El reloj, las señales, las LUTs y las señales de entrada y salida generales son potencia dinámica

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.106 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26,2°C**
Thermal Margin:	58,8°C (4,9 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



En la siguiente imagen se expone el resumen de tiempos. Si vamos punto por punto de este análisis de tiempos, observamos que el WNS (**W**orst **N**egative **S**lack) refleja el tiempo de espera más largo desde que llegue la señal hasta que esta pueda ser capturada de manera segura, en este caso el tiempo es positivo con un valor de 8 nanosegundos, por otro lado, el WHS (**W**orst **H**old **S**lack) refleja el tiempo más largo desde que la señal sea estable hasta que esta pueda volver a ser capturada de manera segura

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	8,369 ns	Worst Hold Slack (WHS):	0,454 ns	Worst Pulse Width Slack (WPWS):	4,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1	Total Number of Endpoints:	1	Total Number of Endpoints:	2

All user specified timing constraints are met.

A continuación, se encuentran las explicaciones de los análisis para la tarea opcional de ampliación.

Aquí podemos observar que se usan dos LUTs más para la lógica y tres registros más, uno para el *flip-flop*, y los otros dos para los *latches*

Site Type	Used	Fixed	Available	Util%
Slice LUTs	6	0	41000	0.01
LUT as Logic	6	0	41000	0.01
LUT as Memory	0	0	13400	0.00
Slice Registers	4	0	82000	<0.01
Register as Flip Flop	2	0	82000	<0.01
Register as Latch	2	0	82000	<0.01
F7 Muxes	0	0	20500	0.00
F8 Muxes	0	0	10250	0.00

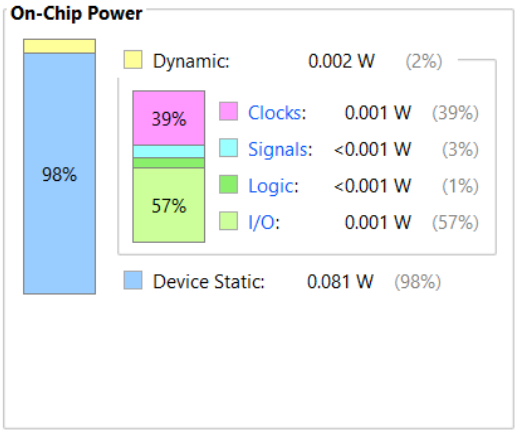
En la imagen a continuación se observa el diagrama de potencia utilizada por el chip, una vez más vemos como nuestro diseño apenas utiliza potencia, ya que el 98% de toda la potencia suministrada al chip es potencia estática, la potencia dinámica es de apenas 0.002W

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.083 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25,2°C**
Thermal Margin:	59,8°C (31,6 W)
Effective ϑJA:	1,9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



La explicación de los datos está en el apartado anterior, por ello simplemente comentaré los resultados. De nuevo todas las restricciones de tiempos han sido respetadas, al observar que todos los *Total Slacks* (TNS, THS, TPWS) tienen valores nulos; se observa poca variación respecto del reporte de tiempos comentado en el apartado anterior.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8,952 ns	Worst Hold Slack (WHS):	0,386 ns	Worst Pulse Width Slack (WPWS):	4,650 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1	Total Number of Endpoints:	1	Total Number of Endpoints:	3

All user specified timing constraints are met.

4. Conclusión y comentarios finales

Este trabajo es entendido como una aproximación muy simplificada a lo que es diseñar un circuito en un entorno profesional. La lógica detrás de una máquina expendedora no es complicada, pero evaluar todos los aspectos detrás de tu circuito tal y como hemos hecho demuestra que diseñar un circuito tiene muchísimas más partes que sólo desarrollar como el circuito debe comportarse, y que son precisamente la optimización de tiempos, potencia y recursos las que diferencian los circuitos aceptables de los circuitos excelentes.