Universidad de Costa Rica

Tarea 3

IE-0523 Circuitos Digitales II

Prof. Enrique Coen Alfaro

Amy Herrera Mora
B53473

19 de mayo de 2024

Índice

1. Resumen

El objetivo principal de la presente tarea era diseñar el controlador para un cajero automático y luego realizar el proceso de síntesis utilizando la herramienta de síntesis Yosys y una biblioteca de elementos lógicos proporcionada en clase. En primer lugar, se realizó el diseño conductual de esta tarea, su funcionamiento comienza cuando el usuario introduce una tarjeta válida, activando una señal y actualizando el PIN correspondiente. El usuario ingresa los dígitos del PIN uno a uno, con cada ingreso activando una señal STB. Tras introducir los cuatro dígitos, el sistema compara el PIN ingresado con el de la tarjeta. Si es incorrecto, se activa una señal de error y se incrementa un contador de intentos fallidos. Se activa una advertencia tras dos intentos inválidos y se bloquea el sistema cuando se llega a tres, requiriendo un reinicio. Si el PIN es correcto, el sistema verifica el tipo de transacción: para depósitos, el tipo de transacción debe estar en bajo, la señal de MONTO activa y procede al estado de déposito donde se suma el monto al balance y se actualiza la señal correspondiente; por otra parte, para retiros el tipo de transacción debe estar en alto, la señal de MONTO activada y se verifica si el monto es menor o igual al balance disponible. Si es así, se actualiza el balance y se ordena la entrega del dinero. De lo contrario, se enciende una señal de fondos insuficientes. Una vez que funcionara correctamente, se seleccionó la biblioteca cmos_cells.lib para realizar el mapeo tecnológico del diseño. Dicho proceso garantizó la correcta síntesis, verificación y evaluación del controlador del cajero automático, utilizando una combinación de herramientas de síntesis y bibliotecas específicas. El sistema demostró su eficacia y viabilidad al obtener de forma exitosa una descripción estructural de lo que fue la descripción conductual de este diseño. Como principales conclusiones se tiene que el sistema funcionó y demostró su viabilidad para ambas descripciones al diseñar por partes, y que los problemas encontrados durante el proceso, como las señales desfasadas que no permitían una correcta transición de estados y la comparación de dos registros de distinta cantidad de bits, fueron superados con recursos disponibles.

2. Descripción Arquitectónica

2.1. Diagrama de bloques

Para realizar el diseño de esta tarea se hizo uso del diagrama de bloques expuesto en la figura (1). Aquí se observan las entradas en la parte izquierda que se le ingresaran al controlador, y las salidas que se desean de acuerdos a dichas entradas en la parte derecha ambas tanto para la versión conductual como para la estructural según la sugerencia del enunciado.

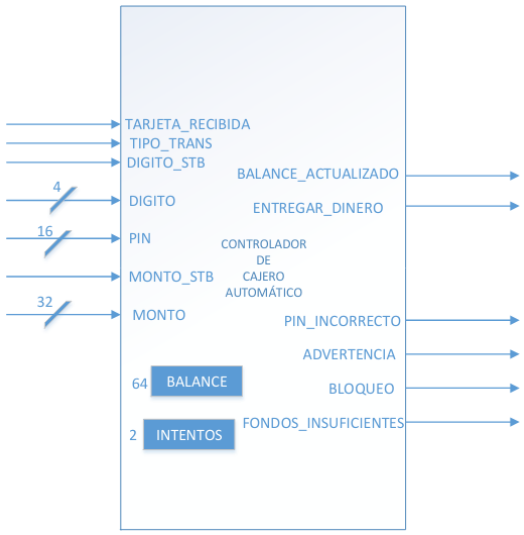

Figura 1: Diagrama de bloques para el controlador de la Tarea 3

2.2. Codificación e implementación de la Máquina de Estados

Una vez analizado el diagrama de bloques, se obtuvieron 6 estados para implementar este controlador y se detallan a continuación de acuerdo a su codificación: ESTADO_INICIAL = 0000, ESTADO_ESPERANDO_DIGITOS = 0001, ESTADO_VERIFICACION = 0010, ESTADO_AUTORIZADO = 0011, ESTADO_DEPOSITO = 0011, ESTADO_RETIRO = 0101 y ESTADO_BLOQUEO = 0110. Con la figura (2) se explica el diagrama de estados y las transiciones que se quiere que esta Máquina de Estados cumpla con el fin de obtener un correcto funcionamiento.

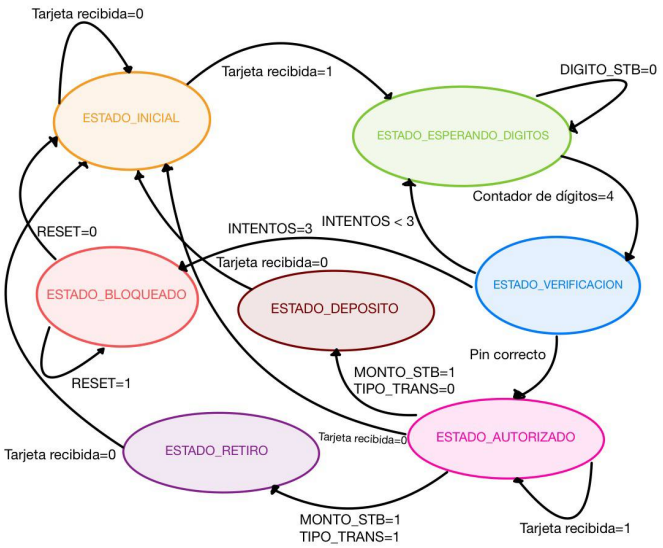

Figura 2: Diagrama de estados para el controlador de la Tarea 3

En esta Máquina de Estados implementada con un bloque *always*@(∗) en Verilog, se inició con un bloque case que depende del estado actual del sistema y se procedió a describir la transición siguiendo

el diagrama de estados anteriormente mencionado. Dicha transición se enumerará acontinuación:

1. **ESTADO_INICIAL:** En este estado, el sistema está en espera. Si la señal de tarjeta recibida está activada, pasa al estado de esperar digitos para comparar el pin en cada intento. De otra manera permanece en este estado hasta que haya una tarjeta recibida.

2. **ESTADO_ESPERANDO_DIGITOS:** En este estado, el sistema espera la entrada de los dígitos del PIN. Cada vez que se ingresa un dígito, el contador de dígitos se incrementa. Cuando se han ingresado cuatro dígitos, se incrementa el contador de intentos, se reinicia el contador de dígitos y el sistema cambia al estado de verificación del PIN.

3. **ESTADO_VERIFICACION:** En el estado de verificación, el sistema compara el PIN ingresado con el PIN interno. Si el PIN es incorrecto, se activa una señal de PIN incorrecto. Si se han hecho dos intentos de ingresar el PIN incorrecto, se activa una advertencia y el sistema vuelve al estado de espera de dígitos. Si se han hecho tres intentos incorrectos, el sistema pasa al estado de bloqueo. Si el PIN es correcto, el sistema pasa al estado autorizado.

4. **ESTADO_AUTORIZADO:** En el estado autorizado, el sistema establece un balance inicial. Dependiendo del tipo de transacción, el sistema decide si se trata de un depósito o un retiro. Si es un depósito y se recibe la señal correspondiente, el sistema cambia al estado de depósito. Si es un retiro y el monto a retirar es menor que el balance, el sistema cambia al estado de retiro. Si el monto a retirar es mayor o igual al balance, se activa una señal de fondos insuficientes y el sistema permanece en el estado autorizado.

5. **ESTADO_DEPOSITO:** En el estado de depósito, el sistema suma el monto del depósito al balance y marca que el balance ha sido actualizado. El sistema permanece en este estado hasta que la tarjeta sea retirada, momento en el cual vuelve al estado inicial.

6. **ESTADO_RETIRO:** En el estado de retiro, el sistema resta el monto a retirar del balance, entrega el dinero y marca que el balance ha sido actualizado. Al igual que en el estado de depósito, el sistema permanece en este estado hasta que la tarjeta sea retirada y luego vuelve al estado inicial.

7. **ESTADO_BLOQUEO:** Si se han hecho tres intentos incorrectos de ingresar el PIN, el sistema activa una señal de bloqueo y permanece en este estado hasta que se active una señal de reinicio, momento en el cual vuelve al estado inicial.

Para la parte del contador, los intentos de ingreso de PIN, la acumulación de cada DIGITO ingresado y la transición de estados se implementó un bloque *always* @(*posedge clk*, por lo tanto se hace una revisión cada flanco de reloj positivo para que el sistema funcione de manera correcta. De acuerdo a los valores que se inicializan en las entradas del diagrama de bloques, obtendremos valores en las salidas cada flanco positivo de reloj al estar trabajando con Flip Flops. Con esto se finaliza la descripción arquitectónica para la versión conductual y la versión estructural de este controlador de cajero automático.

3. Plan de Pruebas

En la tabla (1) se puede observan el Plan de Pruebas mínimo para garantizar el corrector funcionamiento de las descripciones conductual y estructural para esta Tarea. En ella, se hace referencia al número de prueba, si falló o no, y la descripción de cada una en orden. Dichas pruebas fueron implementadas con un único archivo tester.v para ambas versiones. Esto se hizo en un bloque de initial donde se cambiaban los valores de las entradas para verificar el valor de las salidas con la herramienta gtkwave.

Prueba	**Falló**	**Descripción**
1	No	Prueba de señales de ADVERTENCIA, PIN_INCORRECTO y BLOQUEO al ingresar tres pines incorrectos haciendo uso de DIGITOS, DIGITO_STB y de la variable interna INTENTOS.
2	No	Prueba de ingreso de pin correcto para evaluar el funcionamiento de realizar un depósito haciendo uso de las señales de tipo de transferencia, MONTO, MONTO_STB y el BALANCE interno.
3	No	Prueba de ingreso de dos pines incorrectos y luego un pin correcto para observar el buen funcionamiento de las señales de la prueba 1 y del ESTADO_RETIRO cuando MONTO es menor o igual al valor del BALANCE.
4	No	Prueba de señal de FONDOS_INSUFICIENTES al ingresar un pin correcto, un tipo de transferencia y un MONTO mayor al valor del BALANCE.

Tabla 1: Descripción de las pruebas realizadas.

4. Instrucciones de utilización de la simulación

Al extraer el archivo B53473.zip se encontrarán 2 directorios. En primer lugar se encuentra *conductual_version* que contiene la descripción conductual del controlador **cajero.v**, el **testbench.v** y el **tester.v**; y para su debida compilación y simulación se debe escribir en la terminal de linux desde el dir B53473 **cd conductual_version** seguido de un **make**. De igual manera se utiliza para el directorio que realiza la síntesis mapeada a la biblioteca CMOS *cmos_version*. Dicho directorio contiene los archivos pertinentes para la síntesis completa del controlador y la simulación de las pruebas al escribir en la terminal desde el dir B53473 **cd cmos_version** seguido de un **make** para observar el funcionamiento.

5. Resultados y Análisis

En la presente tarea se realizó el diseño de un controlador para cajero automático. En primer lugar, se hizo la descripción conductual y se aplicaron las 4 pruebas necesarias para analizar su comportamiento y garantizar su correcto funcionamiento. Una vez obtenido este, se procedió a realizar la síntesis para obtener ahora la descripción estructural de este controlador. El objetivo principal era implementar los comandos de la tabla (2) para realizar la síntesis, y luego poder simular el comportamiento del controlador para verificar si funcionaba igual que la descripción conductual.

Comando en Yosys	Funcionalidad
read_verilog name_file.v	Lee el archivo que contiene la descripción conductual en Verilog. Convierte los if y los else en árbol de decisión.
hierarchy -check -top -controller	Se utiliza cuando existen varios módulos y necesito establecer una jerarquía
proc	Convierte los procesos a netlists. Depura el árbol de decisión y construye muxes. Sabe la cantidad de componentes que necesitará.
opt	Se utiliza después de cada comando de síntesis para realizar una optimización. Quita celdas vacías e idénticas para simplificar.
fsm	Comando utilizado para detectar una máquina de estados desde la descripción conductual.
memory	Se utiliza por si existen memorias
techmap	Este comando convierte a la biblioteca genérica de Yosys. Se utiliza para mapear a una tecnología. Genera la versión RTLIL de la síntesis.
dfflibmap -liberty ./cmos_cells.lib	Este comando mapea los flip flops al asociar al archivo liberty para que sepa el tipo de FlipFlop, pues este contiene las características físicas de cada celda.
abc -liberty ./cmos_cells.lib	Con el comando de abc se convierte la lógica booleana a la biblioteca de componentes que se quiere usar donde se proporcionan las características físicas de las celdas con las que se quiere trabajar específicamente.
show	Este comando se puede utilizar cada vez que se mapea y se va realizando la síntesis paso a paso para visualizarlo en forma de cables y compuertas.
write_verilog synth_name_file.v	Este comando se utiliza para escribir en forma de código la síntesis. Se puede obtener la síntesis tanto para el mapeo de la biblioteca genérica de Yosys como para la biblioteca cmos que queremos usar.
clean	Utilizado para realizar una limpieza.

Tabla 2: Tabla de comandos para realizar la síntesis con la herramienta Yosys.

Se procedió a realizar la síntesis utilizando los comandos que mapean a una biblioteca CMOS llamada *cmos_cells.lib* que consta de la caracterización de un buffer, un inversor, una compuerta NAND, una compuerta NOR y un FlipFlop con su área y sus respectivos pines. Al realizar un resumen en la (3) se presentan la cantidad de compuertas obtenidas para poder realizar este controlador, así como la cantidad de FlipFlops necesarios.

Cantidad de compuertas obtenidas en la síntesis	
NOT cells	67
NAND cells	144
NOR cells	144
Flip-Flops D	25

Tabla 3: Tabla de componentes

5.1. Comportamiento del controlador de cajero automático

5.1.1. Prueba 1: Ingreso de tres pines incorrectos y luego bloqueo de sistema

Cuando la entrada de tarjeta ingresada se pone en 1, paso del estado inicial a esperando dígitos. Una vez ingresados los cuatro dígitos, paso al estado de verificación e INTENTOS se incrementa en 1 porque existen intentos válidos e inválidos. Si la comparación del pin no es correcta, mi intento se vuelve inválido, levanto la señal de pin incorrecto y vuelvo a esperar más dígitos. Una vez que obtengo dos intentos inválidos, levanto la señal de advertencia y vuelvo a esperar más digitos porque el límite de intentos inválidos es 3. Si llego a tres intentos inválidos, paso al estado de bloqueo donde se levanta la señal de bloqueo y sólo vuelvo al estado inicial si hago un RESET del controlador del cajero. En la figura (3) se puede apreciar el correcto funcionamiento para la descripción conductual y en la figura (4) se observa este mismo comportamiento pero ahora en la descripción estructural.

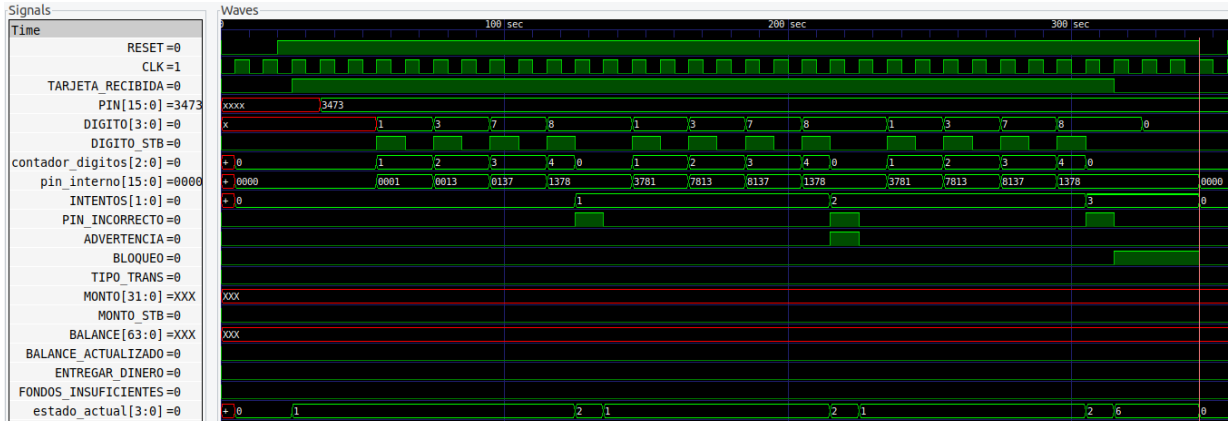

Figura 3: Simulación del correcto funcionamiento de la descripción conductual para la prueba 1.

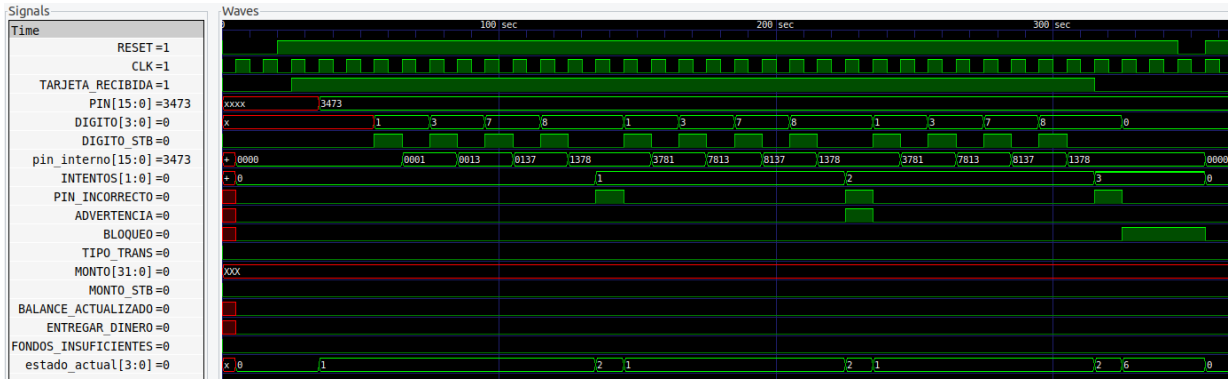

Figura 4: Simulación del correcto funcionamiento de la descripción estructural para la prueba 1.

5.1.2. Prueba 2: Ingreso de un pin correcto y luego depósito de dinero

Cuando la entrada de tarjeta ingresada se pone en 1 de nuevo, paso del estado inicial a esperando dígitos. Una vez ingresados los cuatro dígitos y almacenados en pin interno, paso al estado de verificación e INTENTOS se incrementa en 1 porque existen intentos válidos e inválidos. Al ser pin correcto paso al estado autorizado y obtengo cierto BALANCE asociado a la tarjeta ingresada para proceder de acuerdo al tipo de transacción, que para este caso está en bajo lo que significa que es un depósito y paso al estado depósito. Al tener un MONTO y la señal de MONTO activa simplemente hago la suma BALANCE = BALANCE + MONTO y levanto la señal de balance actualizado. Terminada la transacción paso al estado inicial. En la figura (5) se puede apreciar el correcto funcionamiento para la descripción conductual y en la figura (6) se observa este mismo comportamiento pero ahora en la descripción estructural.

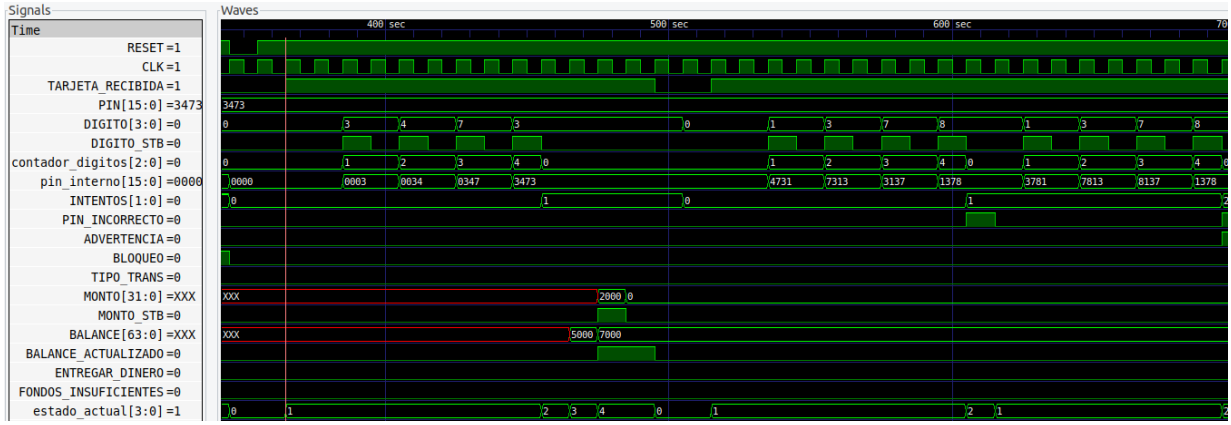

Figura 5: Simulación del correcto funcionamiento de la descripción conductual para la prueba 2.

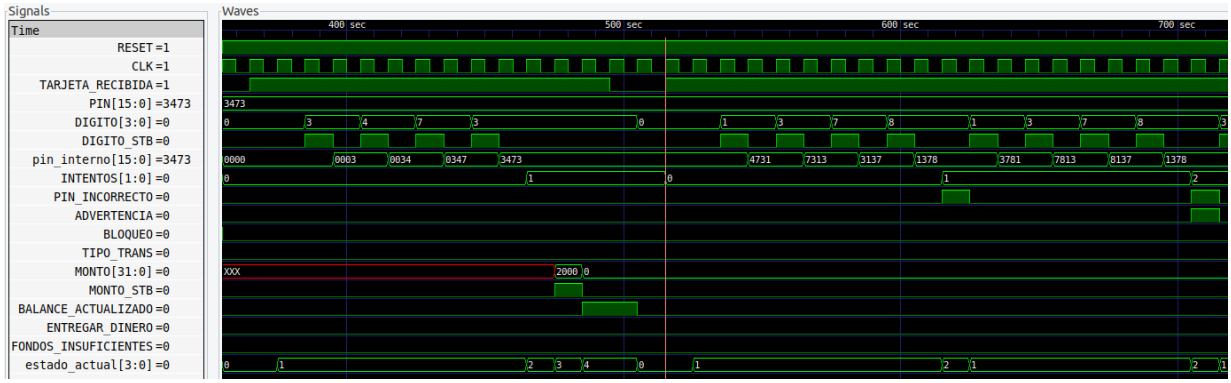

Figura 6: Simulación del correcto funcionamiento de la descripción estructural para la prueba 2.

5.1.3. Prueba 3: Ingreso de dos pines incorrectos y luego retiro de dinero

Cuando la entrada de tarjeta ingresada se pone en 1, paso del estado inicial a esperando dígitos. Una vez ingresados los cuatro dígitos, paso al estado de verificación, pero al ser un pin incorrecto, levanto señal de pin incorrecto, incremento en 1 INTENTOS y vuelvo a pedir cuatro dígitos más. Seguidamente, se ingresa un pin incorrecto más, se incrementa INTENTOS EN 2, se levanta la señal de pin incorrecto y de ADVERTENCIA. Se vuelve a esperar el ingreso de 4 dígitos más y al ser este un pin correcto se pasa al estado autorizado. Como el tipo de transacción está en alto, pasamos al estado de retiro, y al tener un $MONTO < BALANCE$ y señal de MONTO activada simplemente se realiza la resta BALANCE = BALANCE - MONTO. Finalmente, actualizo el balance, activo señal de dinero entregado y al siguiente flanco positivo vuelvo al estado inicial. En la figura (7) se puede apreciar el correcto funcionamiento para la descripción conductual y en la figura (8) se observa este mismo comportamiento pero ahora en la descripción estructural.

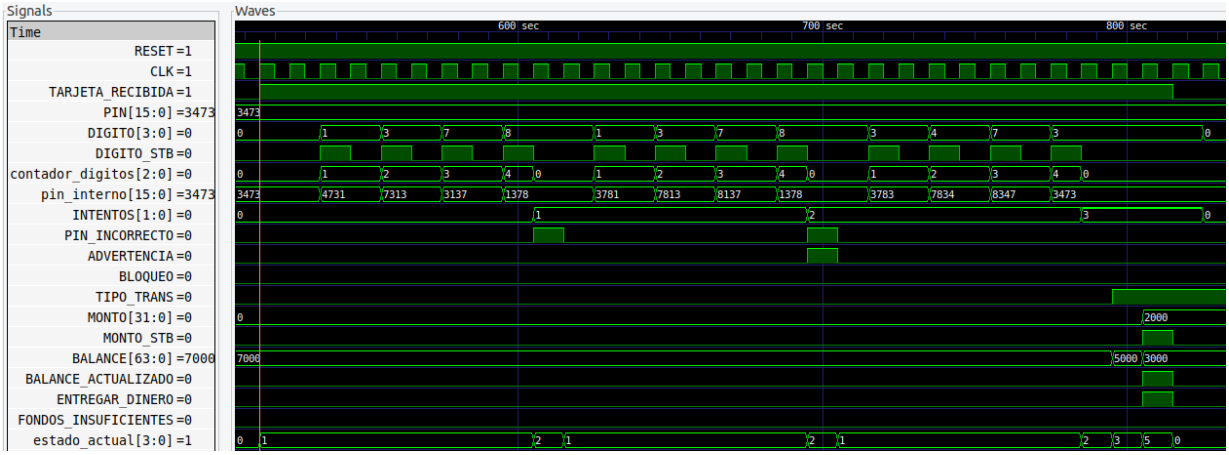

Figura 7: Simulación del correcto funcionamiento de la descripción conductual para la prueba 3.

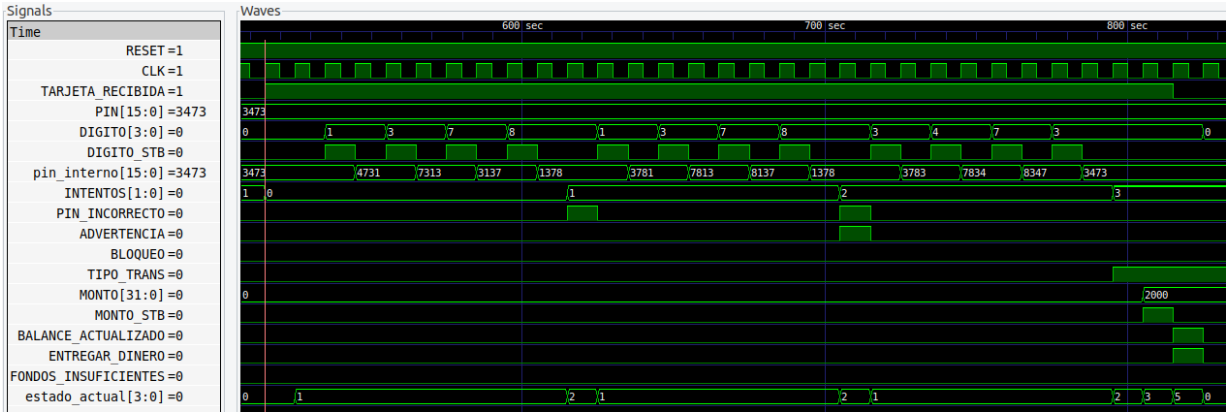

Figura 8: Simulación del correcto funcionamiento de la descripción estructural para la prueba 3.

5.1.4. Prueba 4: Ingreso de pin correcto y intento de retiro pero activación de señal de FONDOS INSUFICIENTES

Cuando la entrada de tarjeta ingresada se pone en 1 de nuevo, paso del estado inicial a esperando dígitos. Una vez ingresados los cuatro dígitos y almacenados en pin interno, paso al estado de verificación e INTENTOS se incrementa en 1 porque existen intentos válidos e inválidos. Al ser pin correcto paso al estado autorizado y obtengo cierto BALANCE asociado a la tarjeta ingresada para proceder de acuerdo al tipo de transacción, que para este caso está en alto lo que significa que es un retiro. Al tener un MONTO y la señal de MONTO activa reviso si el $MONTO < BALANCE$, sin embargo, al revisar que el valor del MONTO sobrepasa al BALANCE, se levanta la señal de FONDOS INSUFICIENTES y no se actualiza el balance, sino que se da por terminado el intento y se vuelve al estado inicial. En la figura (9) se puede apreciar el correcto funcionamiento para la descripción conductual y en la figura (10) se observa este mismo comportamiento pero ahora en la descripción estructural.

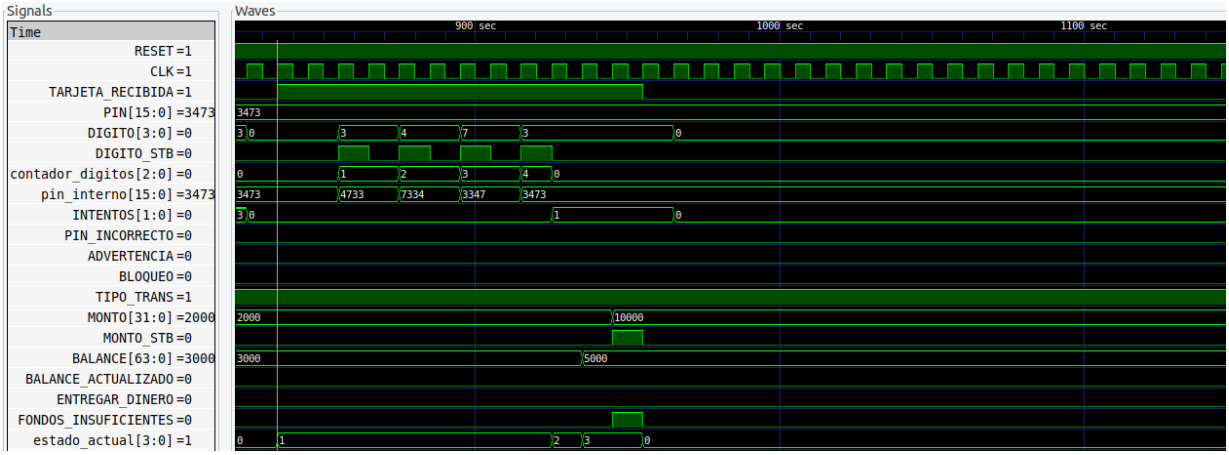

Figura 9: Simulación del correcto funcionamiento de la descripción conductual para la prueba 4.

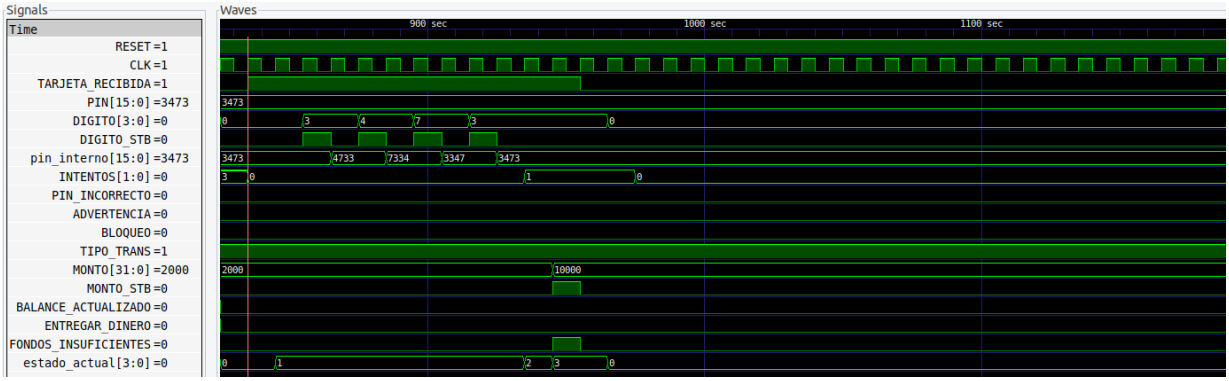

Figura 10: Simulación del correcto funcionamiento de la descripción estructural para la prueba 4.

Al realizar la comparación entre las descripciones conductual y estructural se puede observar que funcionan ambas de manera esperada a diferencia de que en la estructural se nota mejor la transición o valor de las variables en cada flanco positivo de reloj. Este resultado es de suma importancia, pues en la industria parte de los aspectos más relevantes a evaluar en un circuito es su área y el consumo de potencia; y cuando se implementan controladores interesa que realicen su funcionamiento, pero que también se puedan construir en la vida real, y esto sólo se garantiza con la síntesis de un diseño. Además, el hecho de que se haya logrado una síntesis con compuertas específicas dadas logra la minimización buscada por un proceso de síntesis donde existen limitaciones para poder realizar una descripción estructural con lo que se tenga disponible.

6. Conclusiones y Recomendaciones

6.1. Conclusiones

1. El sistema funcionó según lo previsto y demostró su viabilidad con éxito tanto para la descripción conductual como para la descripción estructural. Realizar un diseño paso a paso y realizar simulaciones en cada parte permitió un buen desarrollo de esta tarea.

2. Las simulaciones realizadas en el archivo tester.v fueron todas exitosas y consistentes para cada una de las descripciones. Esto revela que el controlador efectivamente puede convertirse en un circuito físico.

3. Los problemas encontrados durante el proceso, como las partes de la descripción conductual que no se podían sintetizar o señales desfasadas que no permitían una correcta transición de estados, fueron superados con recursos disponibles. Las horas consulta de clase, la documentación, los archivos de clase sobre máquinas de estado en Verilog y la práctica fueron fundamentales para el éxito alcanzado en esta tarea.

4. Se identificaron dos desafíos particulares: uno con el desfase de las señales que no permitía una buena transición de estados y otro con la comparación de dos registros de 32 y 64 bits. El primero se resolvió modificando la lógica para darle tiempo a cada Flip-Flop de actualizarse en cada flanco de reloj positivo, y el segundo se pudo arreglar haciendo uso de un reg temporal que extendiera a 64 bits el reg de 32 en cada comparación para evitar cualquier tipo de problema.

6.2. Recomendaciones

1. Priorizar la fase de diseño inicial para la descripción conductual, con la creación detallada de diagramas y la definición clara de requisitos y funcionalidades del sistema es fundamental para comenzar con el proceso. Dividir el problema total en pequeñas partes e ir armando el código poco a poco es indispensable, y esto se vuelve mucho más relevante en diseños más grandes y complejos.

2. Probar el controlador de forma conductual una vez terminada cada división del problema total, y luego sintetizar también parte a parte y comprobar su funcionamiento permite arreglar problemas de manera más fácil y rápida, pues no todo código en Verilog es sintetizable, y siguiendo estas pautas se vuelve más sencillo poder detectar errores de lógica.

3. Realizar pruebas exhaustivas del sistema mediante simulaciones en diferentes escenarios y condiciones para garantizar su robustez y fiabilidad. Esto permite observar el funcionamiento de nuestro trabajo para poder identificar mejor las fallas, y así poder corregir cualquier problema que se presente.

4. Pedir ayuda externa, ya sea en horas consulta, es indispensable, pues existen muchos errores de funcionalidad que no pueden ser detectados por una única persona que diseña circuitos digitales.

5. Continuar aprendiendo y practicando el uso de Verilog y otras herramientas de diseño de hardware para mejorar constantemente las habilidades y la eficiencia en futuros proyectos similares.