Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0523 Circuitos Digitales II
I ciclo 2024

Tarea #3 Diseño de un controlador para cajero automático

Oscar Porras Silesky C16042
Grupo 001

Profesor: Ing. Enrique Coen

18 de mayo de 2023

Índice

1. Resumen.

Este proyecto presenta el desarrollo de un controlador de ATM para manejar retiros y depósitos. El sistema se programó en Verilog para gestionar las operaciones de depósito y retiro de dinero.

El rendimiento del controlador se demostró eficazmente al manejar todas las situaciones esperadas, incluyendo la verificación correcta del PIN del usuario y la realización segura de transacciones financieras. También responde adecuadamente al introducir un PIN incorrecto varias veces, activando mecanismos de seguridad como bloqueos temporales y alarmas.

Se realizaron pruebas para asegurar la fiabilidad del sistema en varios escenarios, destacando su capacidad para gestionar correctamente las transacciones y responder a intentos de acceso no autorizados. Estas pruebas confirmaron que el controlador es robusto.

Una característica notable de este diseño es la implementación de un sistema de reloj y el uso de Flip Flops, así como compuertas lógicas NAND, NOR y NOT, que mejoraron la eficiencia y la seguridad del controlador.

2. Descripción Arquitectónica.

2.1. Diagrama de Bloques del controlador.

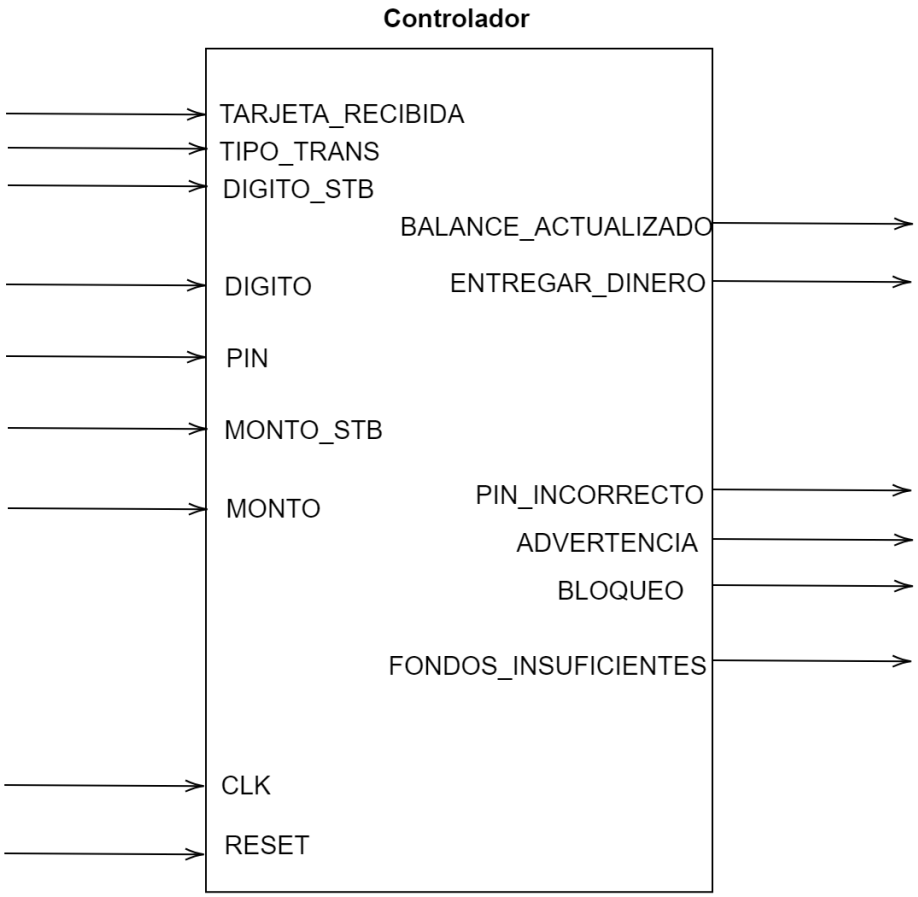

Figura 1: Diagrama de bloques del controlador.

En la figura del controlador, se muestran las interacciones entre las señales del sistema y el módulo `ATMController`. Las señales y su propósito son los siguientes:

- `reset`: Señal utilizada para reiniciar el sistema y restablecer alarmas y contadores de intentos fallidos. Esta señal solo fue utilizada al principio, para un inicio y limpio.
- `clock` (*Clock*): Reloj utilizado para el funcionamiento sincronizado del sistema.
- `PIN`: Entrada de 16 bits que contiene el número de identificación personal (PIN) de la tarjeta, usado para verificar la identidad del usuario.
- `TARJETA_RECIBIDA`: Señal que indica la inserción de una tarjeta en el cajero automático. Activación de esta señal permite al sistema iniciar la verificación del PIN.
- `DIGITO`: Entrada de 4 bits que representa cada dígito ingresado por el usuario del PIN durante la sesión activa.
- `DIGITO_STB`: Señal de pulso que indica que un dígito ha sido ingresado, permitiendo la captura del valor de `DIGITO` por el sistema.
- `TIPO_TRANS`: Entrada binaria que define el tipo de transacción a realizar: 0 para depósito y 1 para retiro.
- `MONTO`: Entrada de 32 bits que especifica el monto a depositar o retirar en la transacción.
- `MONTO_STB`: Señal de pulso que confirma la introducción del monto, permitiendo que el sistema lo procese.
- `BALANCE_ACTUALIZADO`: Salida que se activa cuando el balance de la cuenta ha sido actualizado correctamente después de una transacción.
- `ENTREGAR_DINERO`: Salida que se activa para instruir al cajero automático que entregue el monto especificado durante un retiro.
- `FONDOS_INSUFICIENTES`: Salida que se activa cuando un intento de retiro excede el balance disponible en la cuenta.
- `PIN_INCORRECTO`: Salida que se activa cuando el PIN ingresado no coincide con el PIN asociado a la tarjeta.
- `ADVERTENCIA`: Salida que se activa tras el segundo intento fallido de ingreso de PIN, indicando un aviso antes del bloqueo de la cuenta.
- `BLOQUEO`: Salida que se activa tras el tercer intento fallido de ingreso de PIN, resultando en el bloqueo de la tarjeta y suspensión de todas las operaciones.

2.2. Diagrama de Bloques del funcionamiento de los archivos.

Es importante destacar que `pruebas.v` es el archivo con el módulo en lógica secuencial, `RTLIL.v` es el archivo con el módulo en RTLIL y `synth.v` es el archivo con el módulo sintetizados. Donde `pruebas.v` se puede intercambiar con alguno de los otros dos.

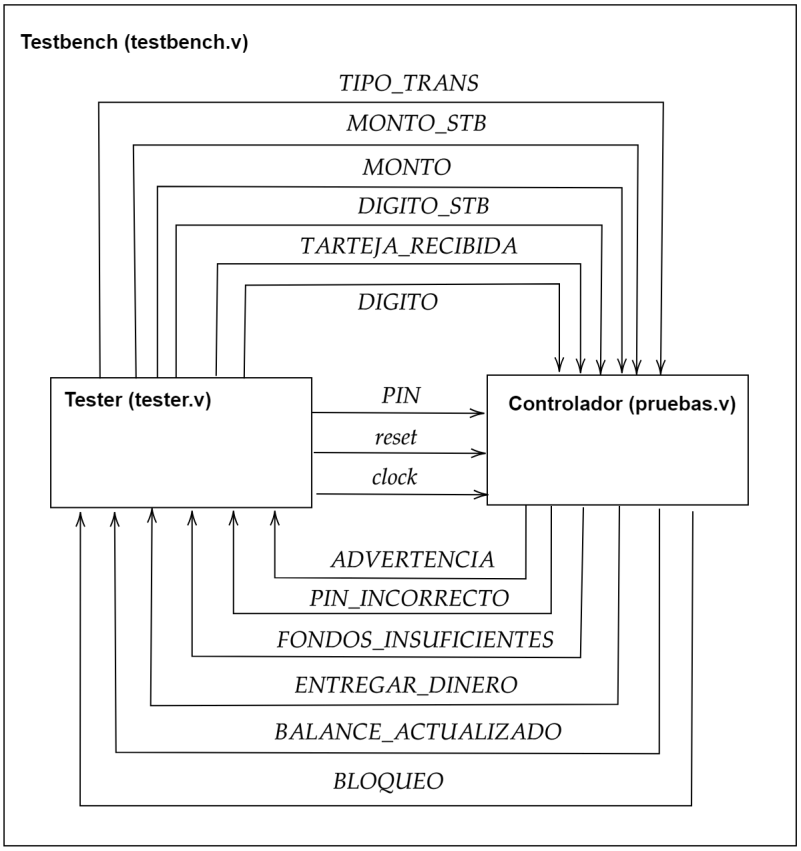

Figura 2: Diagrama de bloques de cómo los archivos se comunican.

2.3. Diagrama de estados.

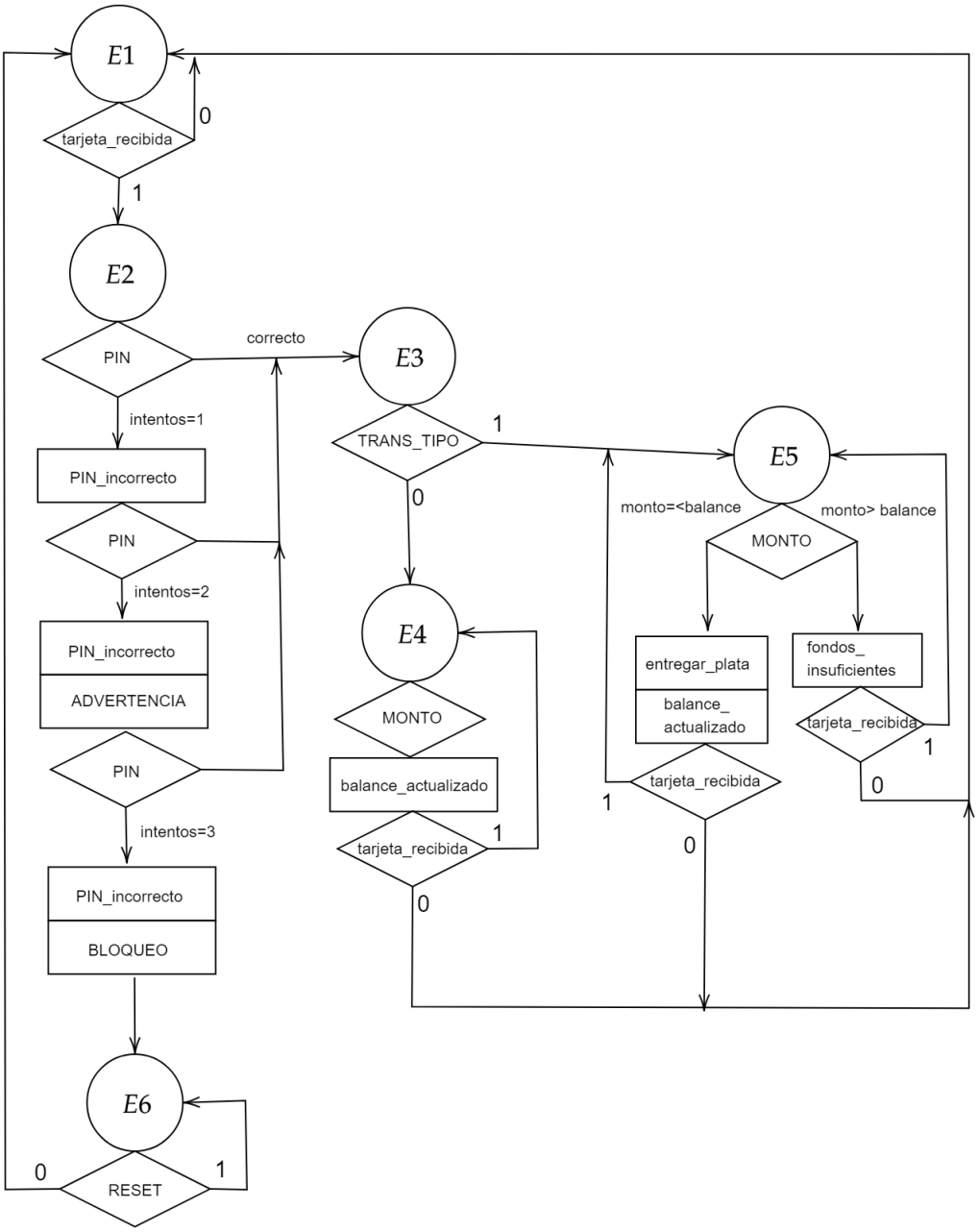

Figura 3: Diagrama de estados del sistema entero.

Los estados del sistema se describen a continuación:

- `IDLE (E1)`: Estado inicial, en espera de que se introduzca una tarjeta. El sistema permanece en este estado hasta que detecta la señal `TARJETA_RECIBIDA` activa.
- `VERIFY_PIN (E2)`: Una vez detectada la tarjeta, el sistema solicita y verifica el PIN introducido por el usuario. Si el PIN es incorrecto, se incrementa el contador de intentos; si es correcto, se avanza al siguiente estado.
- `PROCESS_TRANSACTION (E3)`: En este estado, el sistema espera la selección del tipo de transacción a realizar, ya sea un depósito o un retiro, basado en la señal `TIPO_TRANS`.

- `DEPOSIT (E4)`: Si la transacción es un depósito, el sistema procede a actualizar el balance de la cuenta con el monto ingresado, confirmado por la señal `BALANCE_ACTUALIZADO`.
- `WITHDRAWAL (E5)`: Si el tipo de transacción es un retiro, el sistema procesa la solicitud verificando si los fondos son suficientes y, en caso afirmativo, actualiza el balance y permite la entrega del dinero.
- `BLOCKED (E6)`: Estado de bloqueo activado tras tres intentos consecutivos de ingreso de un PIN incorrecto. El sistema se bloquea hasta que se reinicia con la señal `RESET`.

3. Plan de Pruebas

El plan de pruebas se diseñó para validar el funcionamiento del controlador de ATM. Se enumeran a continuación las pruebas realizadas, describiendo cada una con su propósito específico, los pasos ejecutados, y los resultados esperados.

1. **Prueba #1: Inserción de tarjeta y PIN incorrecto en tres ocasiones sucesivas**
   - *Descripción:* Verificar la respuesta del sistema ante la introducción de un PIN incorrecto una, dos y tres veces consecutivas.
   - *Pasos:*
     i. Se simula la inserción de una tarjeta y se introducen cuatro dígitos incorrectos del PIN una vez, luego dos veces y finalmente tres veces.
     ii. Se observa la activación de las señales de PIN incorrecto, advertencia y bloqueo respectivamente.
   - *Resultado esperado:* El sistema debe identificar correctamente cada intento fallido, activar la señal de PIN incorrecto en el primer intento, advertencia en el segundo y bloqueo en el tercero.
   - *Resultado de la prueba:* Exitoso.
2. **Prueba #2: Depósito de dinero**
   - *Descripción:* Validar el procesamiento correcto de un depósito tras la verificación exitosa del PIN.
   - *Pasos:*
     i. Restablecimiento del sistema.
     ii. Inserción de tarjeta y verificación correcta del PIN.
     iii. Selección del tipo de transacción como depósito y entrada del monto a depositar.
   - *Resultado esperado:* El balance debe actualizarse correctamente reflejando el monto depositado y se debe activar la señal de BALANCE_ACTUALIZADO.
   - *Resultado de la prueba:* Exitoso.
3. **Prueba #3: Retiro de dinero**
   - *Descripción:* Evaluar la funcionalidad de retiro, comprobando que el balance se actualiza adecuadamente si los fondos son suficientes.
   - *Pasos:*
     i. Inserción de tarjeta y verificación correcta del PIN.

II. Selección del tipo de transacción como retiro y entrada del monto a retirar.

- *Resultado esperado:* Si el monto es menor o igual al balance, el balance debe disminuir correctamente y la señal de ENTREGAR_DINERO debe activarse. Si no, la señal de FONDOS_INSUFICIENTES debe activarse.
- *Resultado de la prueba:* Exitoso.

4. **Prueba #4: Retiro con fondos insuficientes**

- *Descripción:* Confirmar que el sistema detecta y maneja adecuadamente un intento de retiro que supera los fondos disponibles en la cuenta.
- *Pasos:*
  I. Inserción de tarjeta y verificación correcta del PIN.
  II. Intento de retiro con un monto que excede el balance actual.
- *Resultado esperado:* La señal de FONDOS_INSUFICIENTES debe activarse, indicando que los fondos son insuficientes para el retiro solicitado.
- *Resultado de la prueba:* Exitoso.

Cada prueba fue ejecutada de manera aislada para verificar las reacciones específicas del controlador y asegurar cobertura total de todos los escenarios posibles. Las simulaciones se realizaron utilizando herramientas estándar como Icarus Verilog para observar el comportamiento del sistema ante cada conjunto de entradas.

4. Instrucciones de utilización de la simulación.

4.1. Makefile.

Para facilitar el proceso de compilación y simulación, se ha proporcionado un `Makefile`. Para utilizar el archivo `Makefile`, este debe estar en la misma carpeta que el resto de archivos, al igual que los archivos `cmos_cells.v`, `cmos_cells.lib`, `synthesis.ys` y `gtkwaveconfig.gtkw`. **No olvidar colocar los barra baja en los nombres a la hora de correr los comandos.**

4.1.1. Para solamente compilar y simular el caso conductual:

Se debe correr el comando `make view_conductual`

4.1.2. Para solamente compilar y simular el caso RTLIL:

Se debe correr el comando `make view_RTLIL`

4.1.3. Para solamente compilar y simular el caso sintetizado:

Se debe correr el comando `make view_synth`

4.1.4. Para compilar y simular todos los casos:

Para ejecutar todos los comandos automáticamente, se debe correr el comando `make` en la terminal. Este comando compilará el testbench y los módulos de diseño, ejecutará la simulación para generar el archivo de ondas y abrirá GTKWave con la configuración predefinida para visualizar los resultados. Es importante recalcar que este comando correrá las 3 simulaciones, solo se le debe dar a la X de cerrar ventana del visor de ondas de GTKWave y correrá la siguiente simulación.

4.1.5. Para limpiar todo:

Luego de revisar las ondas y cuando se desee limpiar el directorio de trabajo, se usa el comando `make clean` para eliminar todos los archivos generados durante la compilación y la simulación.

5. Ejemplos de los resultados.

Los resultados que se van a presentar son las 3 simulaciones en su totalidad, y después se analizarán ciertos detalles de la versión sintetizada.

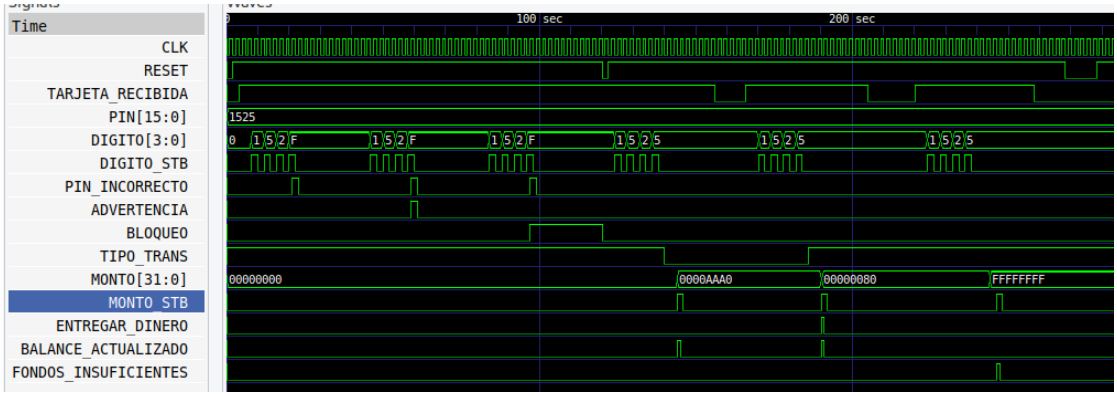

Figura 4: Simulación conductual obtenida con el archivo pruebas.v.

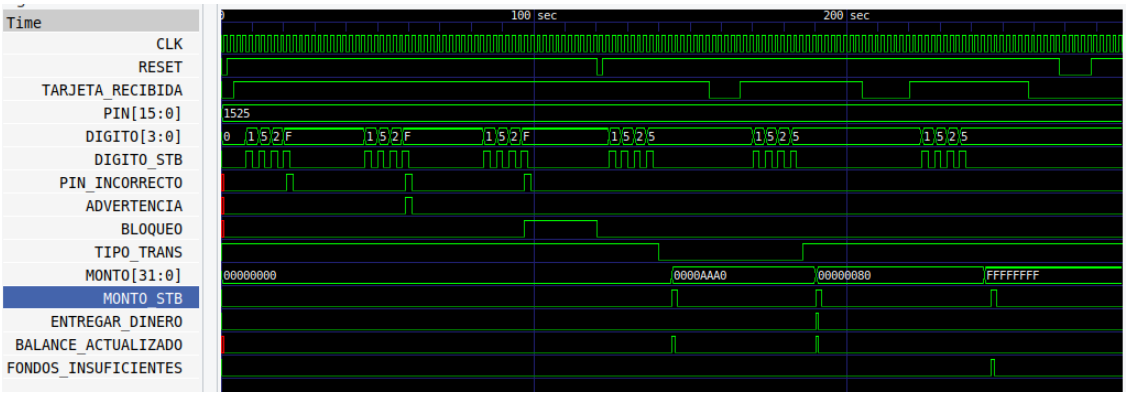

Figura 5: Simulación RTLIL obtenida con el archivo RTLIL.v.

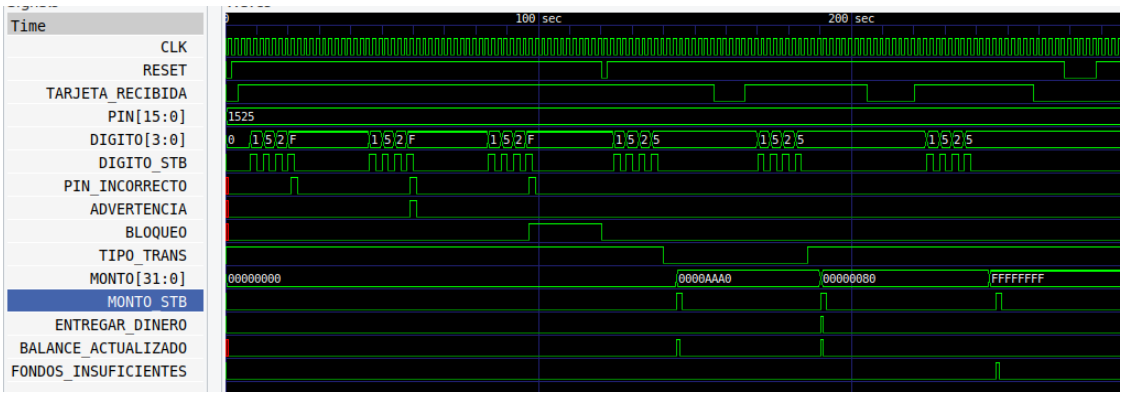

Figura 6: Simulación sintetizada obtenida con el archivo synth.v.

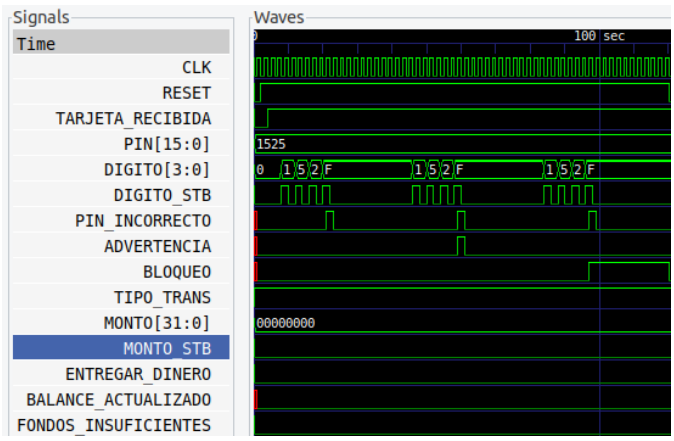

Figura 7: Prueba 1, intentos fallidos con bloqueo.

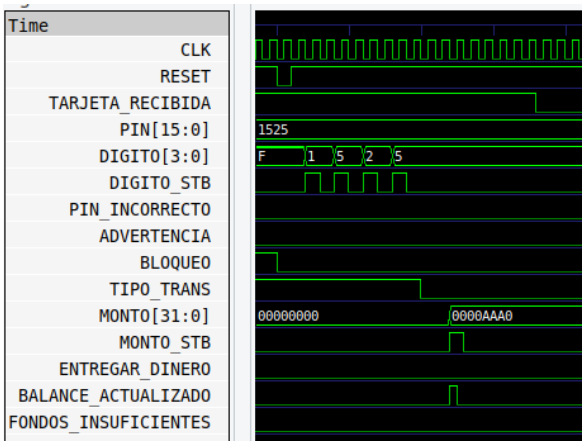

Figura 8: Prueba 2, depósito exitoso.

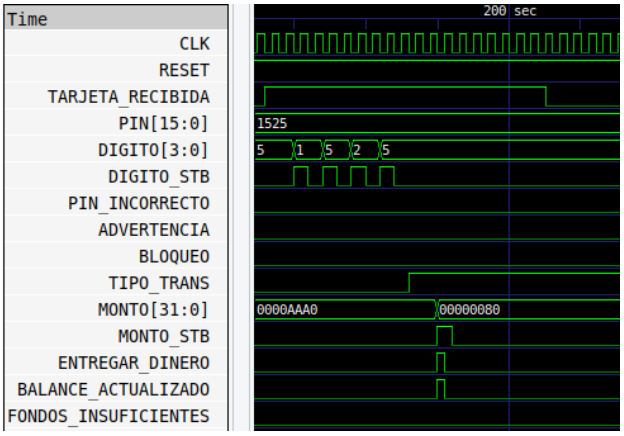

Figura 9: Prueba 3, retiro exitoso.

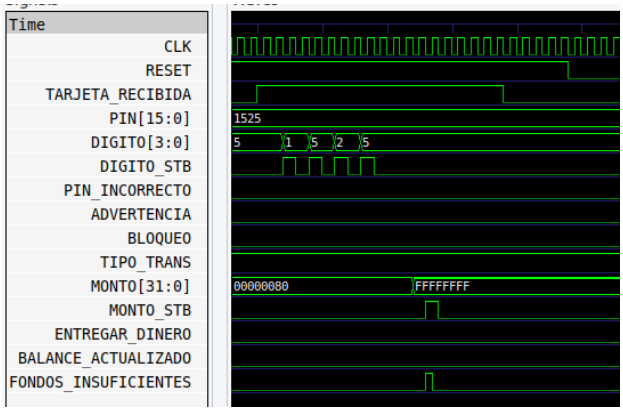

Figura 10: Prueba 4, retiro fallido.

Figura 11: Resultados de las pruebas aplicadas.

Durante la simulación de la primera prueba, se observó la correcta identificación y manejo de entradas de PIN incorrectas sucesivas. En cada instancia (una, dos y tres veces), el sistema activó con precisión las señales de `PIN_INCORRECTO`, `ADVERTENCIA` y `BLOQUEO`, demostrando la eficacia del mecanismo de seguridad implementado para la verificación de autenticidad, como se muestra en la Figura 7.

En la segunda prueba, se comienza con un reset para desactivar el bloqueo, y tras la verificación exitosa del PIN, el sistema procesó adecuadamente un depósito (recordar que la señal `TIPO_TRANS` debe estar en cero). El monto especificado fue agregado al balance de la cuenta y la señal `BALANCE_ACTUALIZADO` se activó, confirmado por la actualización correcta del balance en la cuenta del usuario, verificando así el correcto funcionamiento del sistema en el manejo de depósitos, tal como se evidencia en la Figura 8.

La tercera prueba demostró la capacidad del sistema para gestionar correctamente los retiros (recordar que la señal `TIPO_TRANS` debe estar en uno). Tras la verificación del PIN y confirmar que los fondos eran suficientes, el sistema permitió el retiro del monto deseado, activando la señal `ENTREGAR_DINERO`. Esto indica que el sistema realiza adecuadamente las validaciones de fondos antes de autorizar cualquier transacción de retiro, como se ilustra en la Figura 9.

Finalmente, en la última prueba simulada se abordó el escenario de un intento de retiro con fondos insuficientes. Como se esperaba, el sistema activó la señal `FONDOS_INSUFICIENTES`,

impidiendo la transacción. Esto muestra que el sistema pudo detectar y responder a situaciones donde los montos de retiro exceden el balance disponible, como se documenta en la Figura 10.

6. Detalles, retos y complicaciones durante la solución

El principal reto durante el desarrollo del diseño fue la declaración de variables intermedias, ya que se necesitaban para almacenar los estados temporales y datos intermedios cruciales para la lógica de la máquina de estados. Esto se dio debido a la complejidad de manejar múltiples estados y transiciones entre ellos, donde cada estado depende no solo de las entradas actuales sino también del historial de eventos anteriores. La precisión en la declaración y la sincronización de esas variables era esencial para asegurar la integridad del flujo de datos y evitar errores lógicos que podrían resultar en comportamientos inesperados del sistema.

Un problema principal que se tuvo fue que al intentar correr la versión sintetizada, saltaba un no sé de las variables `MONTO` y `DIGITO`, se solucionó haciendo estas entradas 0 al inicio de la simulación.

7. Evaluación de diseño obtenido

Tipo de Celda	Cantidad
NAND	1206
NOR	1053
NOT	353
DFF	92

Tabla 1: Cantidad de celdas lógicas.

El diseño muestra que, a diferencia de las tareas pasadas, este sistema ocupó muchas más compuertas, lo que refleja la complejidad y las exigencias de manejo de datos y estados en el controlador de ATM diseñado. Aunque el uso elevado de compuertas lógicas puede sugerir una mayor complejidad y posiblemente un mayor consumo de energía, también indica una robustez mejorada en términos de manejo de múltiples estados y redundancias que son cruciales para asegurar la fiabilidad y la seguridad. Este aumento se justifica por la necesidad de implementar una lógica de decisión más sofisticada y por los requerimientos de seguridad que deben cumplirse. La evaluación de las celdas usadas en el diseño también sugiere que la optimización de la lógica fue un factor clave durante el desarrollo.

8. Conclusiones y recomendaciones.

En conclusión, el diseño implementado demostró ser capaz de manejar las tareas asociadas con un sistema de control ATM. La utilización de una cantidad significativa de celdas lógicas, como se refleja en la tabla de componentes, ha garantizado que el sistema pueda procesar una variedad de entradas y estados con alta fiabilidad y seguridad. Sin embargo, esta configuración también implica un consumo de recursos y energía más elevado, lo cual es una consideración a tomar en cuenta.

Las recomendaciones para futuros desarrollos o mejoras incluyen:

- **Optimización de hardware**: Se sugiere revisar el diseño actual con el fin de identificar oportunidades para reducir el número de celdas lógicas sin comprometer la funcionalidad

ni la seguridad del sistema. Esto podría lograrse mediante la integración de funciones lógicas o utilizando tecnologías de síntesis más avanzadas que optimicen el diseño a nivel de hardware.

- **Mejoras en la eficiencia energética**: Dado el alto número de celdas utilizadas, se recomienda explorar alternativas para mejorar la eficiencia energética del sistema, como el uso de celdas de menor potencia o técnicas de gestión de energía dinámica durante las operaciones de bajo movimiento.
- **Expansión de funcionalidades**: Considerar la implementación de nuevas funcionalidades como el pago de servicios, préstamos, entro otros.