Universidad de Costa Rica

Escuela de Ingenería Eléctrica

IE-0523 Circuitos Digitales II

Tarea 3

Profesor:
Enrique Coen Alfaro

Alumno:
Diego Alfaro Segura C20259

5 de octubre de 2024

1. Resumen

En el presente diseño se propone una arquitectura y lógica de unidad de control para un controlador de un cajero. Este sigue el funcionamiento donde se recibe un conjunto de datos de una tarjeta (el pin, balance inicial), un set de instrucciones de la interfaz del cajero (ingreso de pin, monto y tipo de transacción) y se debe realizar la operación. Esto involucra actualizar un registro interno del balance debido a la operación (al igual que notificar que se realizó la operación y si se debe retirar dinero del cajero) y la medición de intentos de ingreso de pin, si se dan intentos incorrectos se debe notificar con 3 señales, una para pin incorrecto, otra de advertencia cuando queda un último intento y la última enunciando que se bloqueó el cajero. Para realizar la tarea se realizó la descripción conductual del módulo en Verilog por medio de una máquina de estados. Para el diseño se empleó un conjunto de contadores y registros internos para cumplir con las especificaciones. Además, se realizó la síntesis del módulo con la librería de celdas cmos dadas por medio del sintetizador Yosys. Finalmente, se realizaron 5 pruebas de funcionamiento las cuales verificaron que el diseño planteado, y su forma sintetizada, cumplen con las condiciones dadas en el enunciado. Durante el proceso se encontró que el manejo de señales de muchos bits, de registros internos y de lógica aditiva (sumas) aumenta considerablemente el tamaño de un diseño sintetizado, al punto que se vuelve difícil generar un diagrama simple. Aún así, se logró diseñar una descripción válida, sintetizable y funcional del módulo pedido.

2. Descripción Arquitectónica

Conforme al diseño general del módulo, se empleó el diagrama de bloques dado en el enunciado, con la única excepción siendo que se agregó la señal "BALANCE_INICIAL" de entrada. Se debe denotar que se hicieron algunas decisiones de diseño. La primera es que se planteó que se debe esperar a que DIGITO_STB se active 4 veces para tener el pin completo y poder compararlo (no comparar dígito por dígito). Este cambio se hizo pues se considera que es más seguro, ya que alguien no autorizado que esté ingresando el pin no podría saber cual dígito es el incorrecto. Para lograr este comportamiento se tiene un registro interno de los dígitos ingresados, este no se muestra en el diagrama pues no es necesario para describir el funcionamiento, pero se muestra en las pruebas como DIGITOS_IN. Otra decisión de diseño fue que las señales PIN_INCORRECTO y ADVERTENCIA se activan solo en el ciclo donde se detecta que se ingresó un pin incorrecto. Esto se hace pensando en una interfaz que despliegue al usuario que ingresó un dígito incorrecto, esta señal no debería mantenerse encendida mientras ingresa los nuevos dígitos pues podría ser confuso (se podría asumir que los nuevos dígitos están mal). En cambio, la señal BLOQUEO se mantiene pues se piensa como una señal que bloquea el cajero y activa algún tipo de alarma. Además, las señales de BALANCE_ACTUALIZADO, ENTREGAR_DINERO, FONDOS_INSUFICIENTES y el cambio de BALANCE se dan solo durante el ciclo del estado respectivo pues se debe retornar al estado inicial, asumiendo que se ingresarán transacciones nuevas o incluso tarjetas nuevas. Finalmente, se asume que el proceso se da en la secuencia ingresar tarjeta -¿digitar pin -¿ingresar monto, por esto se restringe el uso de MONTO_STB a ese punto en la operación y se incluye como un requerimiento para que el módulo siga sus funciones.

Luego, en la Figura 1 se muestra el diagrama ASM realizado para la máquina de estados del cajero. Se usa el formato usual de dicho diagrama, con la excepción que los cambios a contadores/registros internos se muestran como cajas pintadas con gris, esto para diferenciarlos de las salidas.

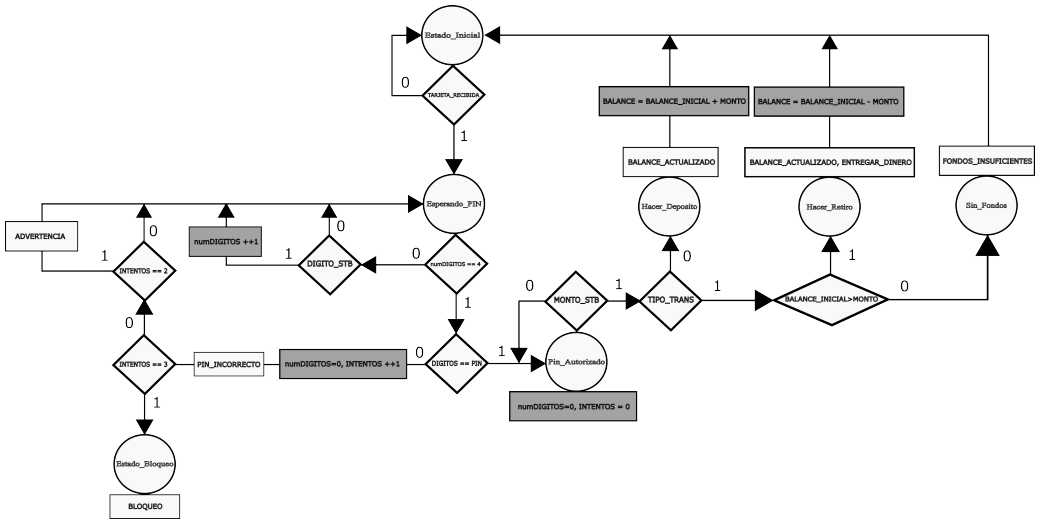

Figura 1: Diagrama ASM de la máquina de estados del cajero.

Los estados asignados son los siguientes:

- Estado_Inicial: Este es el estado inicial, a este estado se llega desde sí mismo si no se ha ingresado una tarjeta, si se da una transferencia (ya sea exitosa o no) o si se aplica la señal de Reset. Se espera a que llegue una tarjeta para así pasar al estado Esperando_PIN.
- Esperando_PIN: Este estado se espera que se ingresen los dígitos a comparar con el pin. Si se ingresa uno (denotado por DIGITO_STB), se actualiza un contador de número de dígitos (numDIGITOS) y se guarda en DIGITOS_IN. Si el contador numDIGITOS llega a ser 4, en este punto se realiza la comparación. Si el pin ingresado es igual al pin de la tarjeta se pasa al estado Pin_Autorizado, sino se eleva la señal de pin incorrecto, y dependiendo de la cantidad de intentos incorrectos se enciende la señal ADVERTENCIA (2 intentos incorrectos) o se pasa al estado Estado_Bloqueo (3 intentos incorrectos).
- Estado_Bloqueo: En este se habilita la señal de BLOQUEO y se permanece en el estado. Solo se puede salir del estado al aplicar el Reset.
- Pin_Autorizado: Ocurre cuando se ingresa el pin correcto. Aquí se hace el check para que se ingrese el monto a usar, si ocurre se pasa a procesar el tipo de transacción y si se puede realizar, sino se mantiene en el estado. Si el tipo de transacción es 0 (osea depósito) se pasa directamente al estado Hacer_Deposito, si es 1 (osea un retiro) se verifica si se tienen los fondos suficientes. Si se tienen los fondos suficientes se pasa al estado Hacer_Retiro, sino se pasa al estado Sin_Fondos.
- Hacer_Deposito: En este se habilita la señal de BALANCE_ACTUALIZADO y se actualiza el registro interno del balance agregándole el monto especificado. Se pasa al Estado_Inicial.
- Hacer_Retiro: En este se habilita la señal de BALANCE_ACTUALIZADO, la de ENTREGAR_DINERO y se actualiza el registro interno del balance restándole el monto especificado. Se pasa al Estado_Inicial.
- Sin_Fondos: En este solo se habilita la señal de FONDOS_INSUFICIENTES. Se pasa al Estado_Inicial.

Los estados se codificaron de la siguiente manera (Tabla 1), con el objetivo de asignar un flip-flop a cada estado y así reducir la posibilidad de que se den condiciones de carrera durante la operación del chip. Se agrega el equivalente de dicha codificación en hexadecimal para tener mayor facilidad de lectura de los estados en los diagramas de tiempo de la Sección 5.

Cuadro 1: Asignación de estados

Estado	Codificación	Equivalente en hexadecimal
Estado_Inicial	0000001	01
Esperando_PIN	0000010	02
Pin_Autorizado	0000100	04
Estado_Bloqueo	0001000	08
Hacer_Deposito	0010000	10
Hacer_Retiro	0100000	20
Sin_Fondos	1000000	40

La estructura del código describe una máquina de estados que emplea flip-flops para actualizar los estados en el flanco creciente de la señal de reloj, al igual que las salidas. Además, se configuró la señal de Reset de tal manera que tome acción en estado bajo según su flanco decreciente.

3. Plan de pruebas

Para verificar el funcionamiento típico se emplearon cuatro tipos de pruebas que cubren las situaciones generales que pueden darse durante su operación normal. Se asignó el PIN correcto como 0259.

3.1. Prueba #1, deposito.

Se comprueba que se puede ingresar el pin correcto, un monto y realizar el depósito. Se espera de esta prueba que se de la transición de estados $Estado\_Inicial \Rightarrow Esperando\_PIN \Rightarrow PIN\_Autorizado \Rightarrow Hacer\_Deposito \Rightarrow Estado\_Inicial$. Además se espera que se actualicen correctamente los contadores de dígito por cada ingreso, que se active la salida BALANCE_ACTUALIZADO y se actualice el balance a la suma del balance inicial y el monto. **Note que las transiciones de estados son para los cambios de estados, no necesariamente representan ciclos individuales.**

3.2. Prueba #2, retiro con montos suficientes

Se comprueba que se puede ingresar el pin correcto, un monto menor al balance inicial y realizar un retiro. Se espera de esta prueba que se de la transición de estados $Estado\_Inicial \Rightarrow Esperando\_PIN \Rightarrow PIN\_Autorizado \Rightarrow Hacer\_Retiro \Rightarrow Estado\_Inicial$. Además se espera que se actualicen correctamente los contadores de dígito por cada ingreso, que se active la salida BALANCE_ACTUALIZADO y ENTREGAR_DINERO y se actualice el balance a la resta del balance inicial y el monto.

3.3. Prueba #3, retiro con montos insuficientes

Se comprueba que se puede ingresar el pin correcto, un monto mayor al balance inicial y que se notifique que no se puede realizar el retiro. Se espera de esta prueba que se de la transición de estados $Estado\_Inicial \Rightarrow Esperando\_PIN \Rightarrow PIN\_Autorizado \Rightarrow Sin\_Fondos \Rightarrow Estado\_Inicial$. Además se espera que se actualicen correctamente los contadores de dígito por cada ingreso, que se active la salida FONDOS_INSUFICIENTES y no se cambie el valor del balance.

3.4. Prueba #4, ingreso de pin incorrecto menos de 3 veces

Se comprueba que se puede ingresar un pin incorrecto menos de 3 veces y posteriormente se puede realizar un deposito sin problema. Se espera de esta prueba que se de la transición de estados $Estado\_Inicial \Rightarrow Esperando\_PIN \Rightarrow PIN\_Autorizado \Rightarrow Sin\_Fondos \Rightarrow Estado\_Inicial$. Además se espera que se actualicen correctamente los contadores de dígito por cada ingreso, que se active la salida PIN_INCORRECTO y posterior a esto se siga el mismo comportamiento al de la prueba 1.

3.5. Prueba #5, ingreso de pin incorrecto 3 veces

Se comprueba que se al ingresar un pin incorrecto3 veces se bloquea el sistema hasta que se reinicie. Se espera de esta prueba que se de la transición de estados $Estado\_Inicial \Rightarrow Esperando\_PIN \Rightarrow Estado\_Bloqueo \Rightarrow Estado\_Inicial$. Este último cambio al estado inicial se debe dar solamente cuando se aplique el Reset. Además se espera que se actualicen correctamente los contadores de dígito por cada ingreso, y cuando se ingrese el tercer intento incorrecto (se debe reflejar en el registro de intentos fallidos) se pase al Estado_Bloqueo donde se activa la señal BLOQUEO.

4. Instrucciones de utilización de la simulación

Para repetir las simulaciones utilizadas se incluye un archivo de Makefile que permite correr los comandos necesarios para realizarlas. Se debe tener instalado las herramientas de software Icarus Verilog (compilador de verilog), GTKWave (visualizador de ondas), Yosys (sintetizador de código Verilog) y GNU-make o mingw Make (comandos para ejecutar un Makefile). Además, se deben tener los archivos incluidos del diseño (tester.v, testbench.v y Cajero.v), el archivo Makefile en el mismo directorio que estos archivos y los archivos de la librería CMOS (cmos_cells.lib y cmos_cells.v) en un subdirectorio llamado "lib". Se incluyen 2 comandos, notar que si se emplea la versión de make de mingw se debe escribir "mingw32-make" en vez de solo "make". **En este caso solo se puede emplear el GNU-make por las restricciones de Yosys**.

Comandos incluidos en el Makefile:

- make simular Corre la simulación al compilar el archivo y mostrar los resultados en GTKWave, está configurado para mostrar todas las ondas importantes según se muestra en los resultados. Este genera la síntesis y la usa en la simulación, pero no muestra el diagrama.

- make full Corre la simulación al compilar el archivo y mostrar los resultados en GTKWave, está configurado para mostrar todas las ondas importantes según se muestra en los resultados. Este si muestra el diagrama resultante. Es posible que se tarde un tiempo considerable para mostrar el diagrama.

- make clean Elimina los archivos generados para limpiar el directorio (no elimina los de código ni el Makefile).

Se hicieron los dos comandos por separado pues el proceso de mostrar el diagrama resultó considerablemente lento, comparado con solo sintetizar y mostrar la simulación. Para que no sea necesario realizar el proceso entero se agrega el pdf del diagrama resultante a los entregables llamado "diagramaSintetizado.pdf".

5. Ejemplos de los resultados

5.1. Resultados de síntesis

Se logró realizar el proceso de síntesis en el módulo diseñado sin problema, así indicando que el diseño final es sintetizable. Se realizó una modificiación en la declaración del registro interno BALANCE, esto pues al sintetizar el módulo se eliminaba dicho registro (y toda su lógica) pues no se utiliza para nada según las especificaciones. Para poder sintetizar el diseño completo y mostrar el registro en la simulación se agregó la línea "(* keep *)" a la declaración del registro, esto le indica a Yosys que no lo debe eliminar. Se presenta una tabla con el resultado final de componentes usados en la síntesis:

Cuadro 2: Componentes de síntesis final.

Celda	Cantidad
NOT	269
NAND	852
NOR	709
DFF	60

A través de las simulaciones se obtuvieron los resultados mostrados en los siguientes diagramas de tiempos. Todos estos provienen de una sola simulación, pero se separa en distintas imágenes, una para cada una de las pruebas realizadas. En general los resultados de cada una de las pruebas cumplieron las expectativas descritas en la Sección 3, incluyendo las transiciones de estados y las salidas generadas. Para los cambios de estados es útil consultar la Tabla 1. Debido a esto solo se resaltan los puntos clave de los resultados.

5.2. Resultados prueba #1

En la Figura 2 se muestra el ingreso de la tarjeta y en este punto se actualizan los datos que son pertinentes a la tarjeta. Luego se da el ingreso de los dígitos de la clave correcta, se nota que al pasar estos se agregaban los dígitos al registro correcto y que se aumentaba el contador de dígitos correctamente. Otro aspecto verificado fue el hecho de que se espera al ingreso del monto de la transacción para realizarla, una vez se aceptó el pin. Además, se visualiza que se cumple la transición de estados esperada y se llega a activar la salida correcta y el registro de BALANCE se actualizó correctamente. Se corrobora que se da la comparación solo hasta que se tienen todos los dígitos del pin ingresado al notar que intentos nunca aumentó en valor.

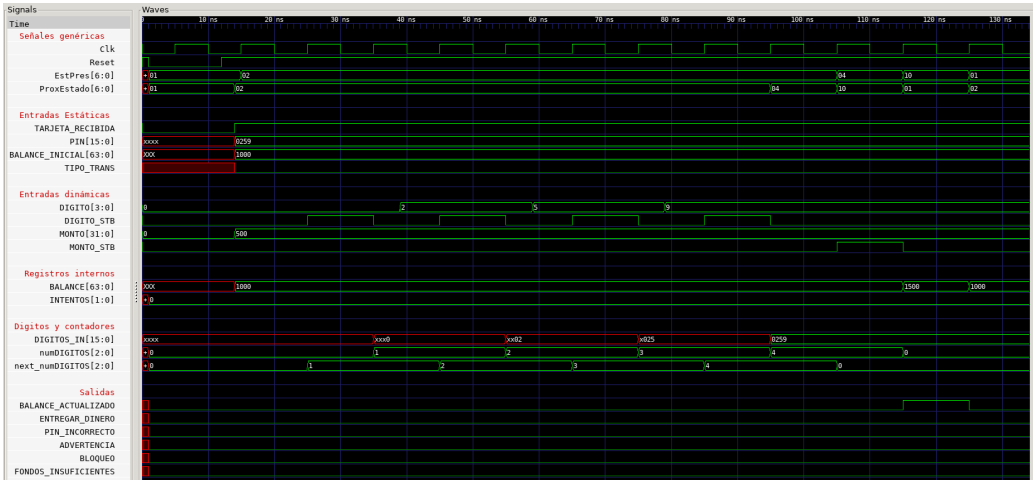

Figura 2: Diagrama de tiempo de prueba #1.

5.3. Resultados prueba #2

En la segunda prueba se obtuvieron resultados casi idénticos a la pasada (como es esperado), pero esta vez se pasa al estado Hacer_Retiro pues se cambió TIPO_TRANS desde el inicio de la prueba. Otro aspecto a mencionar es que se reinició el balance pues no se especificó que se pudiera sobrescribir el balance en la tarjeta dada, y se corre el riesgo de intentar sobrescribir una tarjeta nueva. Se espera que a futuro si se modifica el módulo para hacer algo con el balance se modifique el comportamiento. Se nota que se cumple la transición de

estados esperada en los momentos esperados (cuando se ingresa todo el pin y cuando se ingresa el monto), y se levantaron las salidas esperadas.

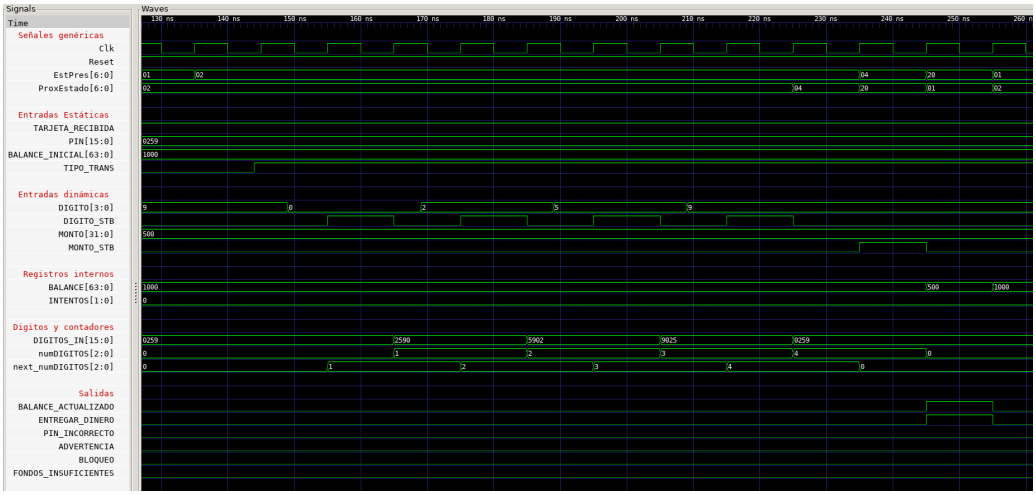

Figura 3: Diagrama de tiempo de prueba #2.

5.4. Resultados prueba #3

En la Figura 4 se muestra la respuesta del sistema ante un retiro con montos insuficientes. Los resultados inicialmente son iguales a los de la prueba pasada, pero debido a que se da un monto muy alto se pasa al estado Sin_Fondos y se activa la señal adecuada. Se nota que no se reescribe el valor del balance y se retorna al estado inicial. Todas las transiciones y salidas fueron según se esperaba.

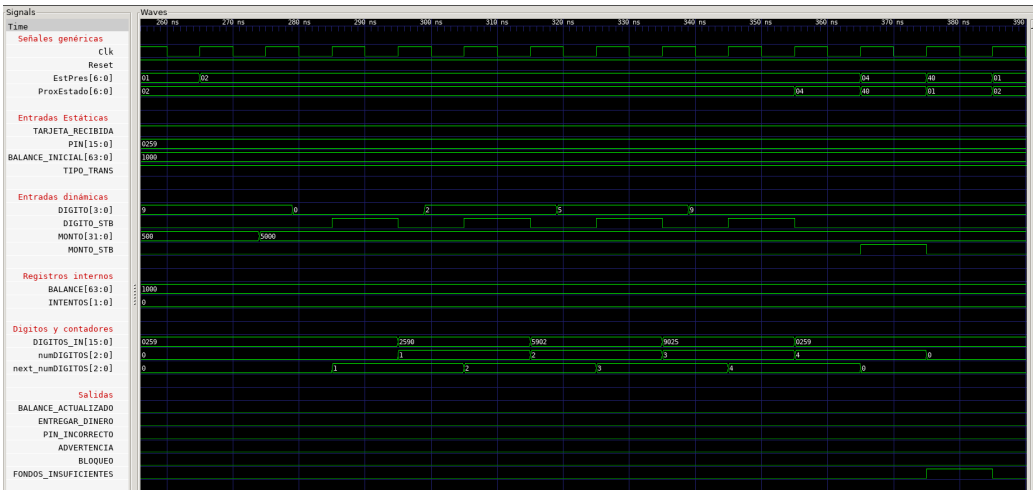

Figura 4: Diagrama de tiempo de prueba #3.

5.5. Resultados prueba #4

En la Figura 5 se muestra la respuesta del módulo ante la entrada de un pin incorrecto. Se nota como se activa la señal de PIN_INCORRECTO solo durante el ciclo en el que se da el pin incorrecto. Además, se logra continuar la transacción posterior a esto sin problema al ingresar el pin correcto. Se puede notar que el contador de intentos se actualizó correctamente y se reinició una vez se volvió al estado inicial. Se cumplieron la transición de estados y las salidas esperadas.

5.6. Resultados prueba #5

En la Figura 5 se muestra la última prueba, donde se ingresa un pin incorrecto 3 veces seguidas, lo cual causa el estado de bloqueo descrito. Se muestra como el contador de intentos fallidos aumenta hasta 3, punto en el cual se pasa al estado de bloqueo, se habilita la señal de bloqueo y se queda ahí hasta que se da la señal de Reset.

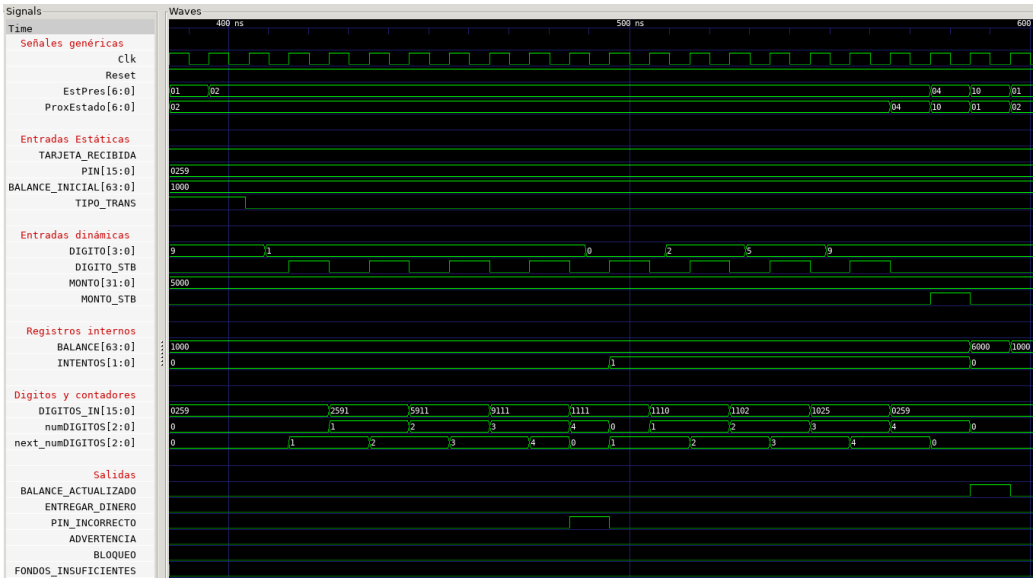

Figura 5: Diagrama de tiempo de prueba #4.

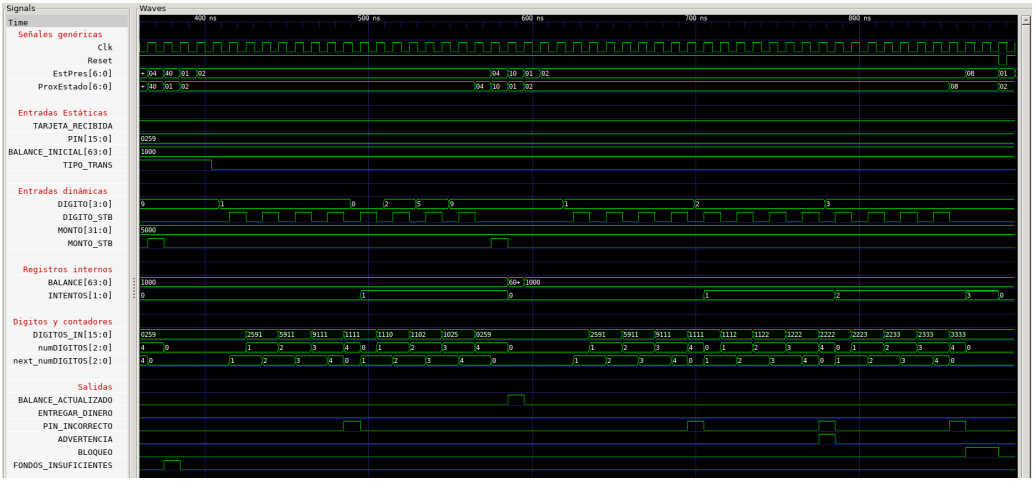

Figura 6: Diagrama de tiempo de prueba #5.

6. Conclusiones y recomendaciones

Durante el proceso de diseño se logró obtener una implementación en Verilog del controlador acorde a las especificaciones dadas y a través de las simulaciones/pruebas se pudo verificar que dicho diseño funciona correctamente. Aún así, se tienen ciertas limitaciones en el diseño, principalmente debido a las entradas/registros grandes del módulo se tiene un tiempo exageradamente extenso para conseguir el diagrama del netlist generado (más no el netlist en sí). Es posible que este tiempo largo sea indicativo de un netlist resultante muy extenso, es difícil de determinar si esto es debido al tamaño normal del módulo a diseñar o debido a una mala optimización durante la síntesis. Se apunta al hecho de que el mismo proceso de síntesis se empleó en la tarea pasada y que los problemas de carga se dieron solo hasta que se incluyó el registro de BALANCE (de 64 bits) como indicativos que los tiempos de carga son debido a la complejidad del sistema y no a una mala síntesis. Conforme a recomendaciones futuras, es pertinente realizar análisis más profundo de la optimización de módulos de esta escala y valdría la pena justificar el uso de señales de tamaño tan elevado, pues podrían no ser necesarias. Se logró aprender del uso de contadores en máquinas de estado, y se categoriza como una forma de solución útil para reducir la cantidad de estados y reducir la complejidad de diseño pero que le deja mucha de dicha complejidad al proceso de síntesis.