Tarea #2 Síntesis de Cajero Automático

Danny Gutiérrez Campos
C33566
Sistemas Digitales II

27/09/2025

1. Resumen

En esta etapa del proyecto, se llevó a cabo la síntesis del diseño conductual del cajero automático digital especificado en la tarea #1. Este proceso generó una descripción estructural (RTLIL), empleando los componentes de la biblioteca interna del sintetizador Yosys. A continuación, se verificó el correcto funcionamiento de dicha descripción estructural utilizando la infraestructura de pruebas desarrollada previamente, asegurando que la secuencia de estados y la lógica del sistema se mantenían según lo esperado.

Posteriormente, se seleccionó la biblioteca `cmos_cells.lib` para realizar el mapeo del diseño. Esto produjo una descripción estructural que utiliza componentes específicos de la biblioteca seleccionada, acercando el diseño a una posible implementación física. Se verificó nuevamente que este diseño mapeado cumpliera con los requisitos funcionales mediante la misma infraestructura de pruebas. Se realizaron modificaciones en los archivos del código `cmos_cells.v` para incorporar retardos temporales en las compuertas lógicas, con el objetivo de evaluar el comportamiento temporal realista del sistema. Tras la modificación, se comprobó que el diseño sintetizado continuaba funcionando correctamente.

Finalmente, se realizó una evaluación del diseño sintetizado considerando:

- Número de componentes utilizados: NAND, NOR, NOT y Flip-Flops (FFs).
- Retardo de propagación desde las entradas hasta las salidas del sistema.

Todas estas acciones se ejecutaron siguiendo el flujo de síntesis definido en el script `Cajero.ys`, que permitió convertir el diseño conductual inicial en una descripción estructural verificada y lista para su análisis final.

2. Plan de Pruebas

A continuación se detallan las pruebas realizadas al diseño del cajero automático utilizando el banco de pruebas `Tester_original.v` y `Tester.v`.

2.1. Tester_original.v

El archivo `Tester_original.v` contiene la infraestructura de pruebas utilizada para validar el correcto funcionamiento del diseño del cajero automático de la tarea #1, es decir, el archivo original sin cambios.

Las pruebas incluidas son:

- **Depósito exitoso**: verifica que el sistema actualiza correctamente el balance tras una operación de depósito válida.
- **Retiro exitoso**: valida que el sistema permite retirar fondos cuando el balance es suficiente y actualiza correctamente el saldo.
- **Bloqueo y desbloqueo**: comprueba que tras varios intentos erróneos de ingreso del PIN el sistema se bloquea, y que posteriormente puede ser desbloqueado según las condiciones especificadas.
- **Fondos insuficientes**: verifica que el sistema detecta y rechaza intentos de retiro cuando el saldo disponible no cubre la operación solicitada.

2.2. Tester.v

El archivo `Tester.v` corresponde a una versión modificada del banco de pruebas original `Tester_original.v`, adaptada para considerar los retardos temporales introducidos en el archivo `cmos_cells.v`. Debido a estos retardos, se incrementó el período del reloj (`CLK`) a un valor de 12 unidades temporales. Para mantener la coherencia de las pruebas y garantizar que las señales de entrada estén alineadas con el ciclo del reloj, se ajustaron todos los valores numéricos de tiempos en las entradas para que sean múltiplos de 12.

Este archivo mantiene los mismos escenarios de prueba definidos originalmente, incluyendo:

- **Depósito exitoso**: Verificación de la correcta actualización del balance tras una operación de depósito válida.
- **Retiro exitoso**: Validación del retiro cuando el balance es suficiente y actualización correcta del saldo.
- **Bloqueo y desbloqueo**: Comprobación del bloqueo del sistema tras intentos incorrectos de ingreso del PIN y la capacidad de desbloqueo posterior.
- **Fondos insuficientes**: Validación de la correcta detección y rechazo de intentos de retiro cuando el saldo disponible es insuficiente.

Este archivo `Tester.v` debería funcionar de manera idéntica a como se describieron todas las pruebas en la tarea #1, ya que la única modificación realizada respecto a `Tester_original.v` corresponde al ajuste del tiempo. El incremento del período del reloj y la modificación de los valores temporales de las entradas fueron implementados únicamente para adecuar las pruebas a los retardos sin alterar la secuencia ni la lógica de los escenarios de prueba. En consecuencia, `Tester.v` garantiza que la verificación del sistema se realiza considerando retardos realistas, ajustando el banco de pruebas a las condiciones temporales del diseño sintetizado, asegurando así una validación robusta y consistente del controlador del cajero automático.

3. Instrucciones de Utilización de la Simulación

Este inciso describe los pasos para ejecutar las simulaciones de la tarea #2. **Recordar que el paso final es solo correr el comando `make` en la terminal dentro del directorio que contenga los siguientes archivos.**

3.1. Paso #1: Archivos Necesarios

Para la correcta utilización de la simulación, es importante organizar todos los archivos en un único directorio. Dicho directorio debe contener exclusivamente los siguientes archivos:

- `Cajero.ys`
- `Cajero_synth.v`
- `Cajero.v`
- `Cajero_rtlil.v`
- `cmos_cells.lib`
- `cmos_cells_retardos.v`
- `cmos_cells.v`
- `Completo.gtkw`
- `Deposito.gtkw`
- `Fondos.gtkw`
- `Bloqueo.gtkw`
- `Retiro.gtkw`
- `Makefile`
- `Testbench.v`
- `Tester_original.v`
- `Tester.v`

3.2. Paso #2: Explicación de uso del Makefile

Dentro del archivo Makefile, la primera parte corresponde a 4 líneas de código comentadas con el fin de que el usuario elija qué archivo y qué versión del plan de pruebas escoger.

```
1 # Selección de archivos fuente a simular:
2 SRC = Cajero_synth.v Tester.v Testbench.v
3 #SRC = Cajero_synth.v Tester_original.v Testbench.v
4 #SRC = Cajero_rtlil.v Tester_original.v Testbench.v
5 #SRC = Cajero.v Tester_original.v Testbench.v
```

Listing 1: Opciones de selección de archivos fuente en la variable `SRC`

En este caso, la variable `SRC` define los archivos fuente que se utilizarán en la simulación. Las alternativas disponibles son:

- `Cajero_synth.v` + `Tester.v`: Versión final sintetizada del cajero, validada con un tester modificado para manejar retardos. Es decir lo requerido por la Tarea #2.
- `Cajero_synth.v` + `Tester_original.v`: Versión final sintetizada del cajero evaluada con el tester original (sin considerar retardos).
- `Cajero_rtlil.v` + `Tester_original.v`: Descripción estructural dada por techmap, comprobada con el tester original.
- `Cajero.v` + `Tester_original.v`: Modelo conductual original del cajero probado con el tester original, es decir lo mismo que en la tarea #1.

Para seleccionar una de estas configuraciones, basta con comentar o descomentar la línea correspondiente en la sección de `SRC`.

Posteriormente en el archivo Makefile viene incluida una sección con el fin de escoger un archivo .gtkw que facilite observar las formas de onda en las pruebas específicas. En donde, el usuario puede comentar y descomentar la línea que desee probar.

```
# Regla por defecto
all:
    iverilog -o $(OUT) $(SRC)
    vvp $(OUT)
    # gtkwave Deposito.gtkw &  #0 a 504
    # gtkwave Retiro.gtkw &  #500 a 936
    # gtkwave Bloqueo.gtkw & #935 a 1980
    # gtkwave Fondos.gtkw &  #1975 a 2574
    gtkwave Completo.gtkw &
# Para probar comente y descomente la sección superior según cada caso
```

Listing 2: Regla por defecto del `Makefile`

- `Deposito.gtkw`: de 0 a 504 s (prueba de depósito exitoso).
- `Retiro.gtkw`: de 500 a 936 s (prueba de retiro exitoso).
- `Bloqueo.gtkw`: de 935 a 1980 s (prueba de bloqueo y desbloqueo).
- `Fondos.gtkw`: de 1975 a 2574 s (prueba de fondos insuficientes).
- `Completo.gtkw`: muestra todas las señales de entradas, salidas y registros internos, sin recortes de tiempo.

Debido a que la simulación siempre genera la traza completa, los números comentados a la par de cada archivo `.gtkw` sirven únicamente como guía para recortar manualmente el intervalo de tiempo dentro de `GTKWave`. Esto permite enfocarse en la parte específica de cada prueba, en caso de que el usuario prefiera analizarla de manera aislada.

3.3. Explicación del Testbench.v

Dentro del archivo `Testbench.v` se encuentra la inclusión de la librería de compuertas. Existen dos versiones disponibles:

- `cmos_cells_retardos.v`: contiene retardos explícitos en cada compuerta.
- `cmos_cells.v`: versión sin retardos.

El en listing 3 se muestra el inicio del `Testbench.v`, en donde, el usuario debe comentar o descomentar la línea correspondiente según cuál versión desee usar:

```
`include "cmos_cells_retardos.v"
/* `include "cmos_cells.v" */
```

Listing 3: Selección de librería de compuertas en el `Testbench.v`

En caso de trabajar con la librería con retardos (`cmos_cells_retardos.v`), es obligatorio que en el `Makefile` se utilice la configuración:

```
SRC = Cajero_synth.v Tester.v Testbench.v
```

Si se selecciona la librería sin retardos (`cmos_cells.v`), entonces se pueden usar cualquiera de las otras tres configuraciones disponibles en `SRC`.

4. Resultados Obtenidos

4.1. Cajero.v sin Retardos

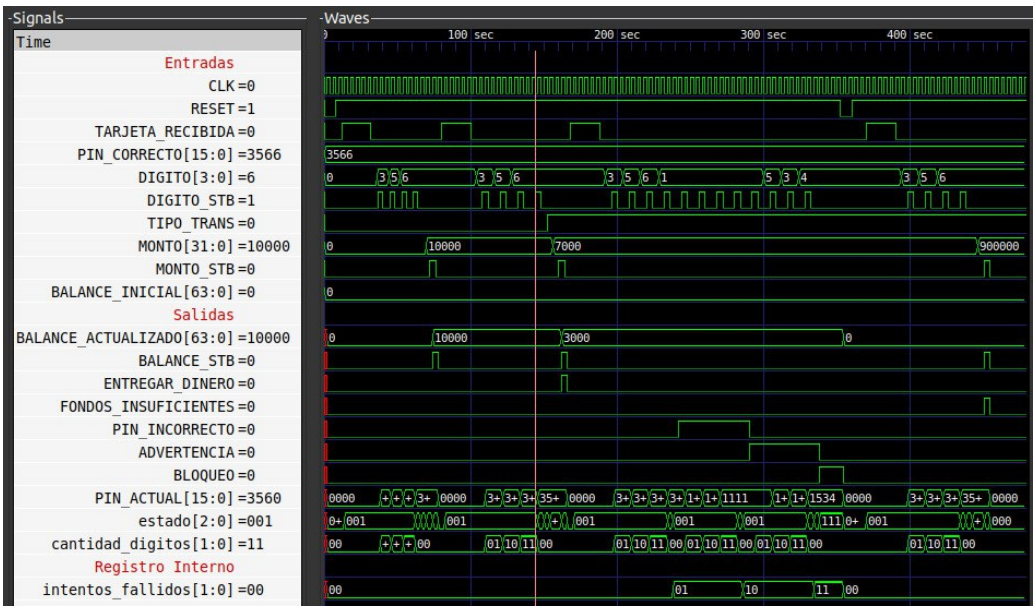

Figura 1: Resultado de la simulación original

Como se observa en la figura 1, esta muestra la simulación completa del Cajero Automático con base en los archivos de la tarea #1. Este resultado muestra cómo se ve la máquina de estados original sin haber pasado por ninguna síntesis ni retardo. Este es el resultado obtenido al correr el Makefile con `SRC = Cajero.v Tester_original.v Testbench.v`, con `gtkwave Completo.gtkw` y el `Testbench.v` con el include `cmos_cells.v`.

4.2. Cajero_rtlil.v sin Retardos

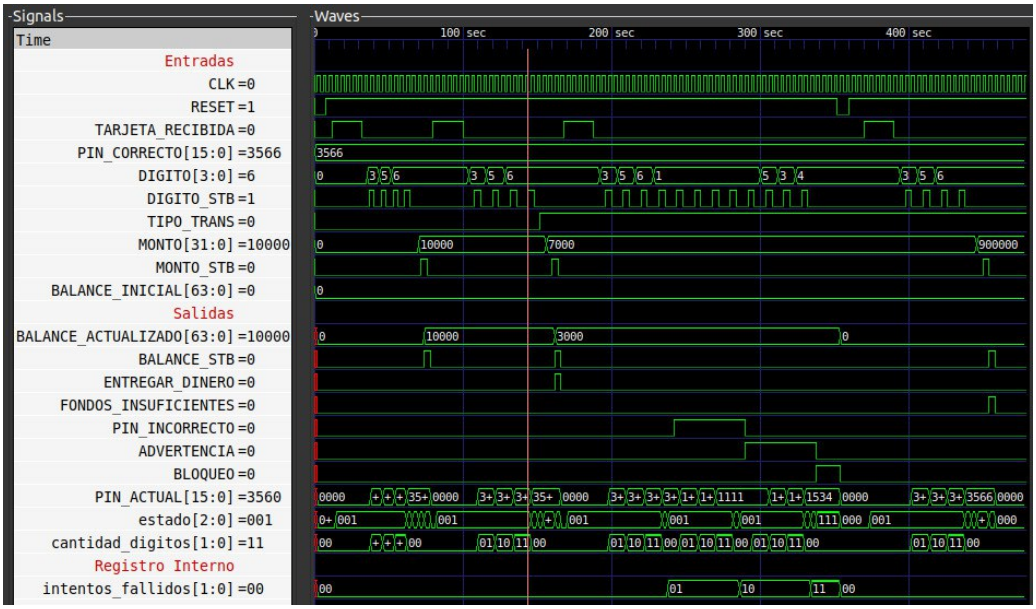

Figura 2: Resultado de la simulación de la síntesis con techmap

Como se observa en la figura 2, esta muestra la simulación completa del Cajero Automático con el archivo `Cajero_rtlil.v`, el cual fue generado justo al haber llegado al paso de (`techmap; opt`) dentro del archivo `Cajero.ys`. Al haber realizado ese paso, se generó una versión sintetizada del archivo original `Cajero.v`. Este resultado muestra que, tal como se esperaba, la máquina de estados funciona exactamente igual al resultado del código original mostrado en la figura 1. Este es el resultado obtenido al correr el Makefile con `SRC = Cajero_rtlil.v Tester_original.v Testbench.v`, con `gtkwave Completo.gtkw` y el `Testbench.v` con el include `cmos_cells.v`.

4.3. Cajero_synth.v sin Retardos

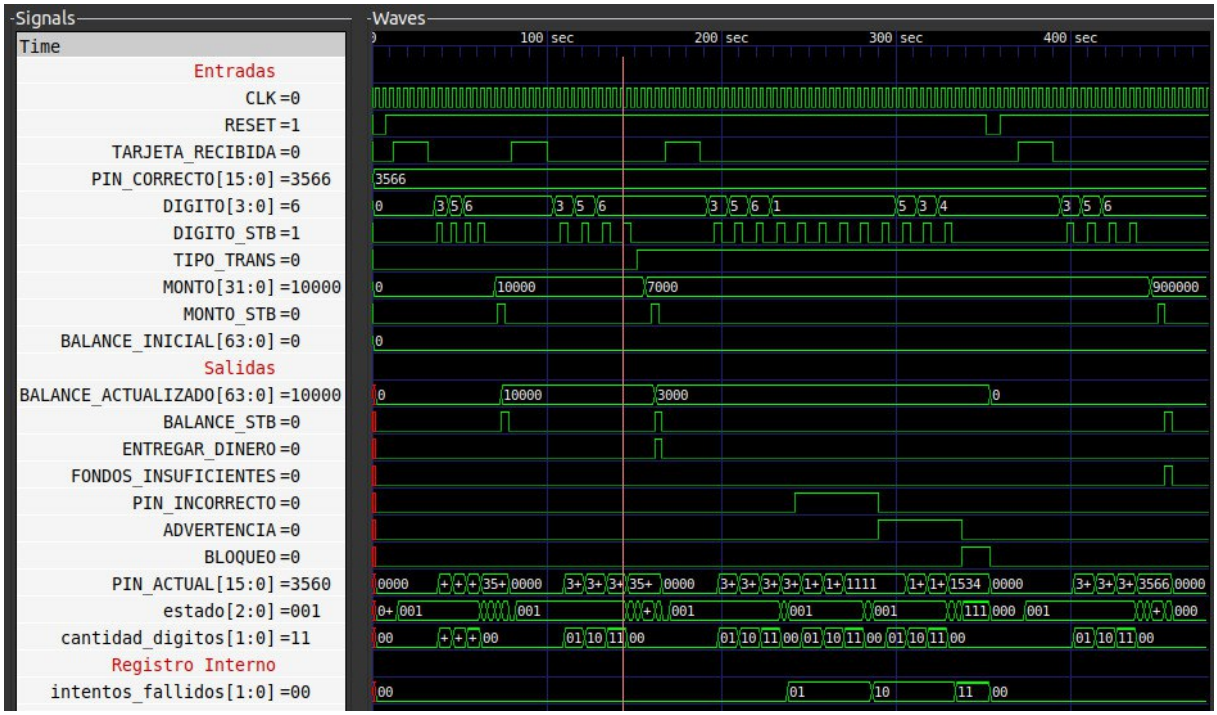

Figura 3: Resultado de la simulación final sin retardos

Como se observa en la figura 3, esta muestra la simulación completa del Cajero Automático utilizando el archivo `Cajero_synth.v`, generado tras completar todos los pasos del archivo Cajero.ys, es decir, con la síntesis final ya realizada. En esta etapa aún no se han agregado retardos, ya que primero se verifica que la síntesis funcione igual que el código original. El resultado mostrado confirma que, como era esperado, la máquina de estados opera de manera idéntica al código original de la figura 1 y a la versión sintetizada hasta `techmap` mostrada en la figura 2. Por lo tanto, se comprueba que la síntesis fue exitosa. Este resultado se obtuvo ejecutando el `Makefile` con `SRC = Cajero_synth.v Tester_original.v Testbench.v`, utilizando `gtkwave Completo.gtkw` y el `Testbench.v` con el `include cmos_cells.v`.

En el código sintetizado del archivo `Cajero_synth.v` generó la siguiente cantidad de elementos lógicos: 1056 compuertas `NAND`, 881 compuertas `NOR`, 385 compuertas `NOT` y 93 flip-flops tipo `DFF`.

4.4. Cajero_synth.v con Retardos

Con la meta de comprobar cómo funciona el sistema ante retardos, se tomó el archivo `cmos_cells.v` y se editó para agregarle retardos a las compuertas, creándose así el nuevo archivo `cmos_cells_retardos.v`. En dicho código se incorporaron retardos de `#1` en las compuertas lógicas (`NOT`, `NAND`, `NOR`) y en el flip-flop tipo D (`DFF`). Al haber hecho este cambio, se realizó una simulación al correr el Makefile con `SRC = Cajero_synth.v Tester_original.v Testbench.v`, con `gtkwave Completo.gtkw` y el `Testbench.v` con el include `cmos_cells_retardos.v`, dando como resultado:

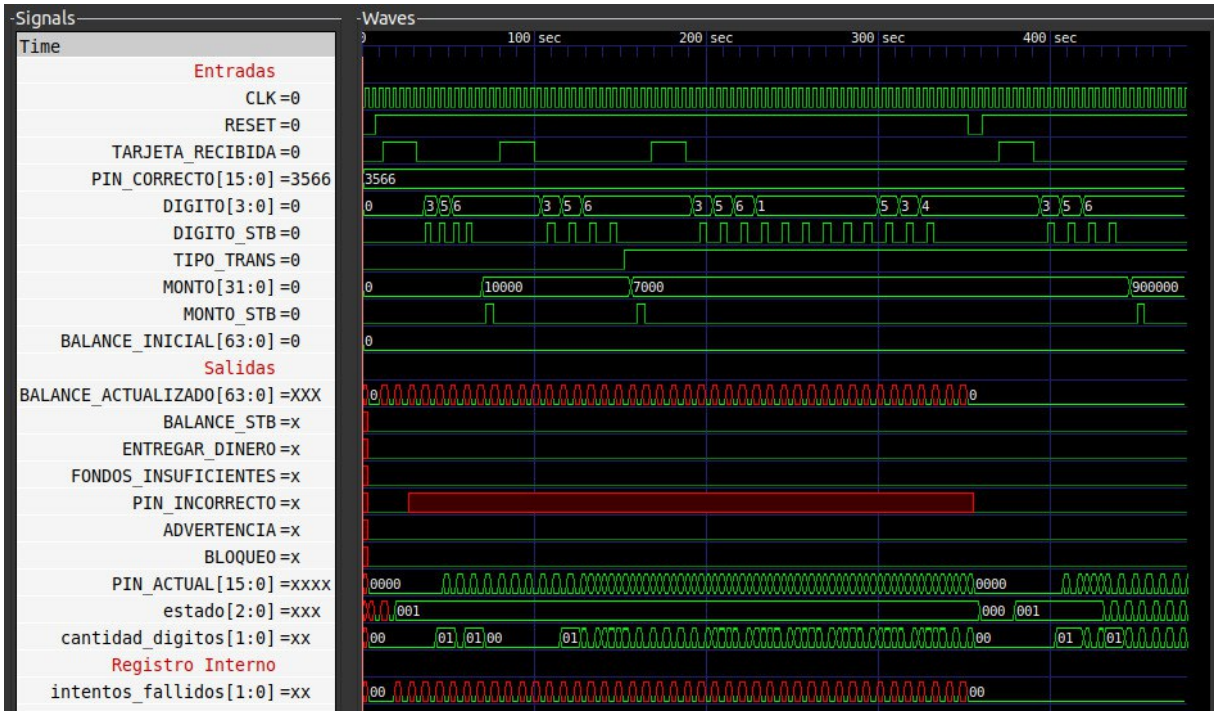

Figura 4: Resultado de la simulación con Retardos - Tester_original

Como era esperable, en la figura 4 el sistema se desconfiguró por completo debido a los retardos agregados en las compuertas. Esto provoca que varias señales nunca lleguen a activarse correctamente, ya que no logran sincronizarse ni entre ellas ni con el reloj. Podemos hacer la analogía con un conjunto de músicos que intentan tocar juntos, si cada uno lleva un pequeño desfase en el tiempo, al final la música se descoordina. De igual forma, en el sistema digital, los retardos provocan desfases que impiden el acoplamiento preciso de las señales con los ciclos del reloj, generando un fallo en la operación del circuito.

Para solucionar dicho problema, se decidió ajustar el archivo encargado de las pruebas, modificando los valores temporales contenidos en él, tal como se explicó en el inciso de la sección 2.2. Este nuevo archivo ajustado recibe el nombre de `Tester.v`. Con base en los cambios implementados en este archivo, se procedió a ejecutar nuevamente la simulación. En esta ocasión, se utilizó el `Makefile` configurado con `SRC = Cajero_synth.v Tester.v Testbench.v`, se cargó `gtkwave Completo.gtkw` para la visualización y se empleó el `Testbench.v` con el `include` de `cmos_cells_retardos.v`. Este conjunto de ajustes permitió que el sistema pudiera sincronizar correctamente las señales con el reloj, produciendo como resultado la configuración mostrada en la figura 5:

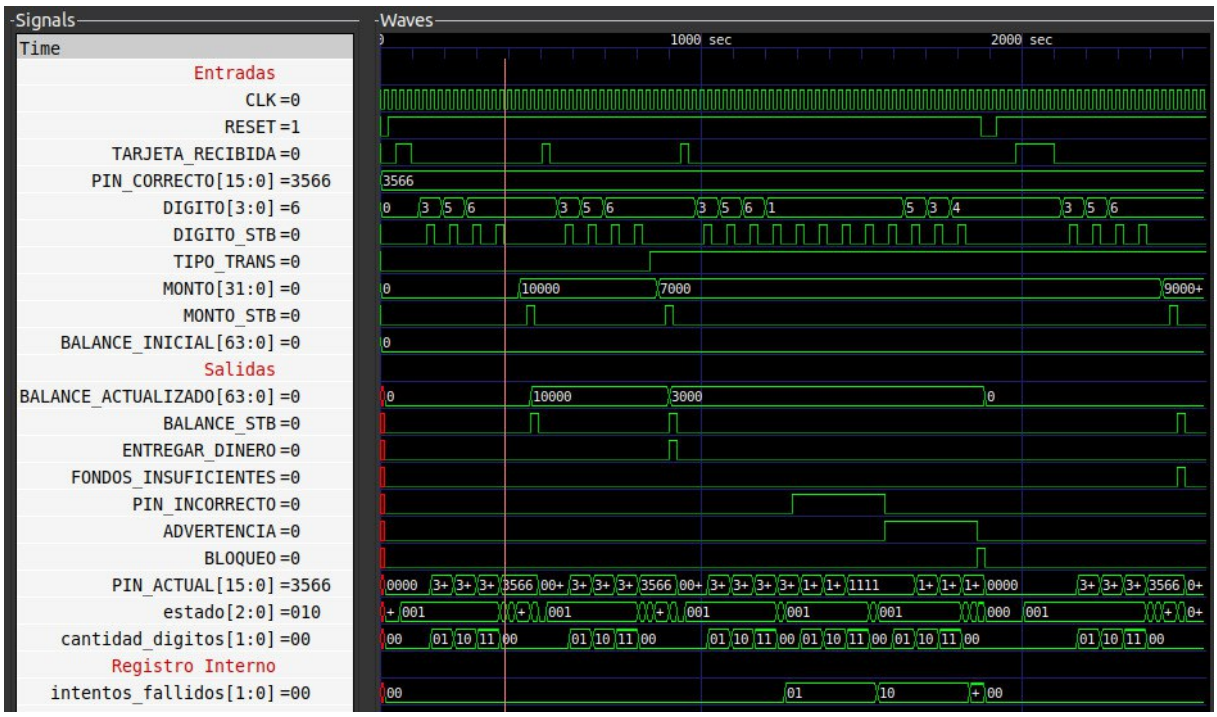

Figura 5: Resultado de la simulación con Retardos - Tester.v

Como se observa en la figura 5, las señales vuelven a funcionar correctamente, y se logró verificar que todo esté correcto, ya que la figura 5 se muestra exactamente igual a las figuras 1, 2 y 3, por ende, funciona correctamente según el plan de pruebas.

4.5. Pruebas Aisladas con Cajero_synth.v y los Retardos Incorporados

Todas estas pruebas se realizan con el `Makefile` con la configuración `SRC = Cajero_synth.v Tester.v Testbench.v` y con el Testbench.v con el include de `cmos_cells_retardos.v`. La finalidad es ver las pruebas por aparte y de cerca, para corroborar que funcionaron.

4.5.1. Prueba Aislada de Deposito

Esta prueba se ejecuta usando `gtkwave Deposito.gtkw` dentro del `Makefile` y recortando la simulación entre 0 y 504 s dentro de GTKWave.

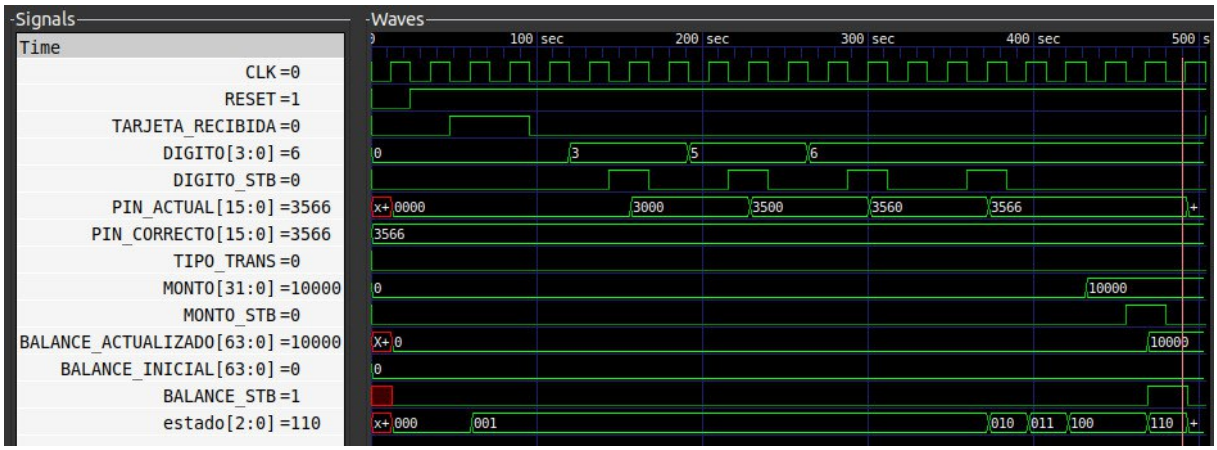

Figura 6: Resultado Simulación de Deposito

4.5.2. Prueba Aislada de Retiro

Esta prueba se ejecuta usando `gtkwave Retiro.gtkw` dentro del `Makefile` y recortando la simulación entre 500 y 936 s dentro de GTKWave.

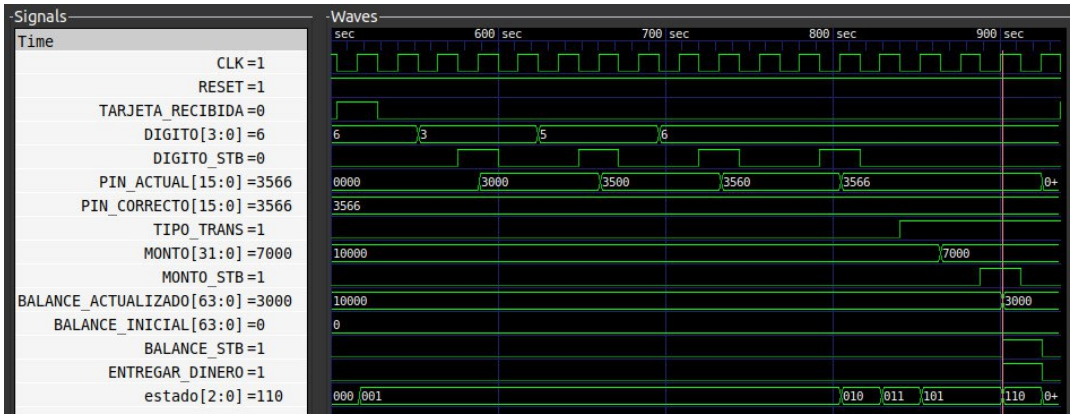

Figura 7: Resultado Simulación de Retiro

4.5.3. Prueba Aislada de Bloqueo y Desbloqueo

Esta prueba se ejecuta usando `gtkwave Bloqueo.gtkw` dentro del `Makefile` y recortando la simulación entre 935 y 1980 s dentro de GTKWave.

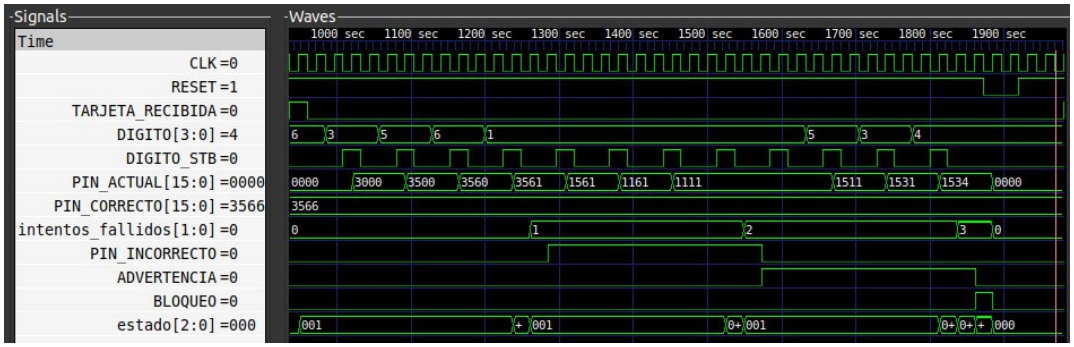

Figura 8: Resultado Simulación de Bloqueo y Desbloqueo

4.5.4. Prueba Aislada de Fondos Insuficientes

Esta prueba se ejecuta usando `gtkwave Fondos.gtkw` dentro del `Makefile` y recortando la simulación entre 1975 y 2574 s dentro de GTKWave.

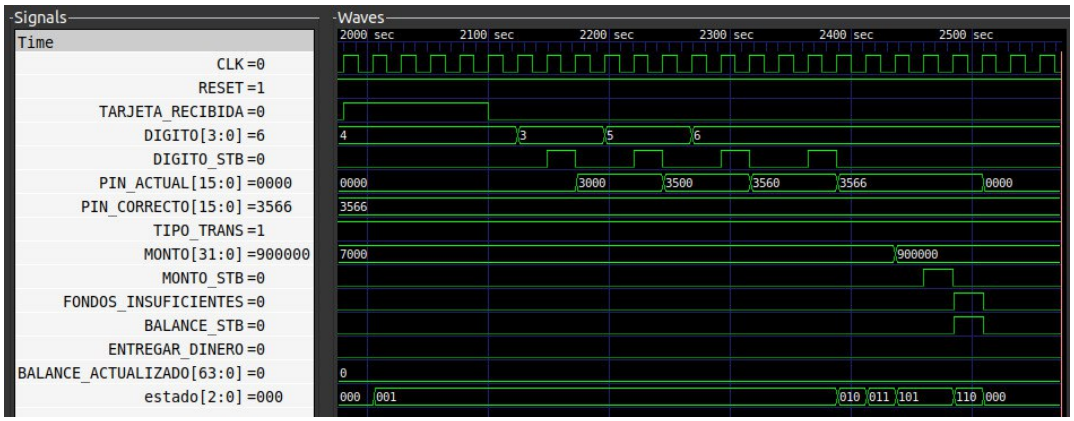

Figura 9: Resultado Simulación de Fondos Insuficientes

5. Conclusiones

En conclusión, el presente proyecto permitió diseñar, sintetizar y verificar un controlador de cajero automático digital, logrando cumplir con los objetivos planteados. Se desarrolló una descripción estructural funcional del sistema y se validó mediante pruebas específicas, incluyendo depósitos, retiros, bloqueo por PIN incorrecto y manejo de fondos insuficientes. Además, se abordó y resolvió exitosamente el reto asociado a los retardos introducidos en las compuertas, adaptando el banco de pruebas y ajustando los tiempos para asegurar que las señales se sincronizaran correctamente con el reloj. Este trabajo demuestra la importancia de considerar los retardos en el diseño digital, ya que, da una idea general de lo que puede ocurrir en la industria real.