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

Tarea 3

Diseño Digital en Verilog: Controlador de Cajero Automatico

Sebastián Bonilla Vega - C01263
Grupo 1

Profesor: Enrique Coen Alfaro

Sábado 18 de Mayo, 2024

Índice

1. Resumen **3**

2. Descripción Arquitectónica **4**

3. Plan de Pruebas **5**

4. Instrucciones de Utilizacion **8**

5. Ejemplos de la Sintesis **9**

6. Conclusiones **10**

1. Resumen

Esta tarea tiene como objetivo desarrollar un circuito digital en Verilog a través de máquinas de estado para representar un controlador de un cajero automático (ATM). Donde su funcionamiento depende VARIAS entradas. Según el enunciado, las entradas son **TARJETA RECIBIDA, TIPO TRANS, DIGITO STB, DIGITO, PIN. MONTO STB y MONTO**. Por otro lado, las salidas del sistema están dadas por **BALANCE ACTUALIZADO, ENTREGAR DINERO, PIN INCORRECTO, ADVERTENCIA, BLOQUEO y FONDOS INSUFICIENTES.** Además, se hace uso de dos variables internas; **BALANCE e INTENTOS.**

- **TARJETA RECIBIDA, TIPO TRANS Y MONTO STB** son ENTRADAS de 1 bit.
- **DIGITO,PIN Y MONTO** son ENTRADAS de 4, 16 y 32 bits respectivamente.
- **BALANCE ACTUALIZADO, ENTREGAR DINERO, PIN INCORRECTO, AVERTENCIA, BLOQUEO Y FONDOS INSUFIENTES** son SALIDAS de 1 bit.
- **BALANCE e INTENTOS** son REGISTROS INTERNOS de 64 y 2 bits respectivamente.

El proyecto se desarrolla a través de una implementación conductual de todo el sistema en VERILOG. Luego, se va a implementar una descripción ESTRUCTURAL del mismo sistema, a través del uso de YOSYS y la librería Cmos.lib. Además, se va a hacer uso de diagramas de bloques y diagramas de estado, realizados a mano. Además, todo el proyecto va a ser desarrollado en Verilog, para reforzar las destresas relacionadas a este lenguaje y para tener más facilidad a la hora de hacer pruebas.

El trabajo llevará a la elaboración de la máquina de estado, demostrando la formación de cada celda a través de flip flops para verificar la contraseña, posterior a ello se realiza el código bajo el lenguaje de Verilog para realizar la simulación de la red. Finalmente se presentarán los resultados mediante las ondas formadas por el sistema en el apartado de GTKWave.

Las pruebas realizadas fueron las sugeridas en el apartado de la tarea:

A. Prueba 1, funcionamiento normal básico con depósito. Se inserta la tarjeta, se lee el pin, se insertan los digitos de manera ideal y sin errores, se comparan con el pin, se selecciona la transacción y se realiza el déposito.

B. Prueba 2, funcionamiento normal básico, con retiro adecuado. Se inserta la tarjeta, se lee el pin, se insertan los digitos de manera ideal y sin errores, se comparan con el pin, se selecciona la transacción y se realiza el retiro.

C. Prueba 3, funcionamiento normal básico, con retiro de montos insuficientes. Se inserta la tarjeta, se lee el pin, se insertan los digitos de manera ideal y sin errores, se comparan con el pin, se selecciona la transacción y se realiza el retiro, pero el monto supera el balance de la cuenta.

D. Prueba 4. Se inserta la tarjeta, se lee pin, pero se insertan digitos incorrectos para levantar ALARMA, PIN INCORRECTO y BLOQUEO. Despues del BLOQUEO, se reinicia todo, se inserta el pin correcto, y se hace un depósito.

E. Prueba 5, prueba de estrés, se levantan advertencias, bloqueo y pines incorrectos. Luego, se hace un depósito. Se inserta la tarjeta, se lee pin, pero se insertan digitos incorrectos para levantar ALARMA, PIN INCORRECTO y BLOQUEO. Despues del BLOQUEO, se reinicia todo, se inserta el pin correcto, y se hace un retiro.

2. Descripción Arquitectónica

En la imagen siguiente, se puede ver el diagrama de bloques propuesto para solucionar el sistema.

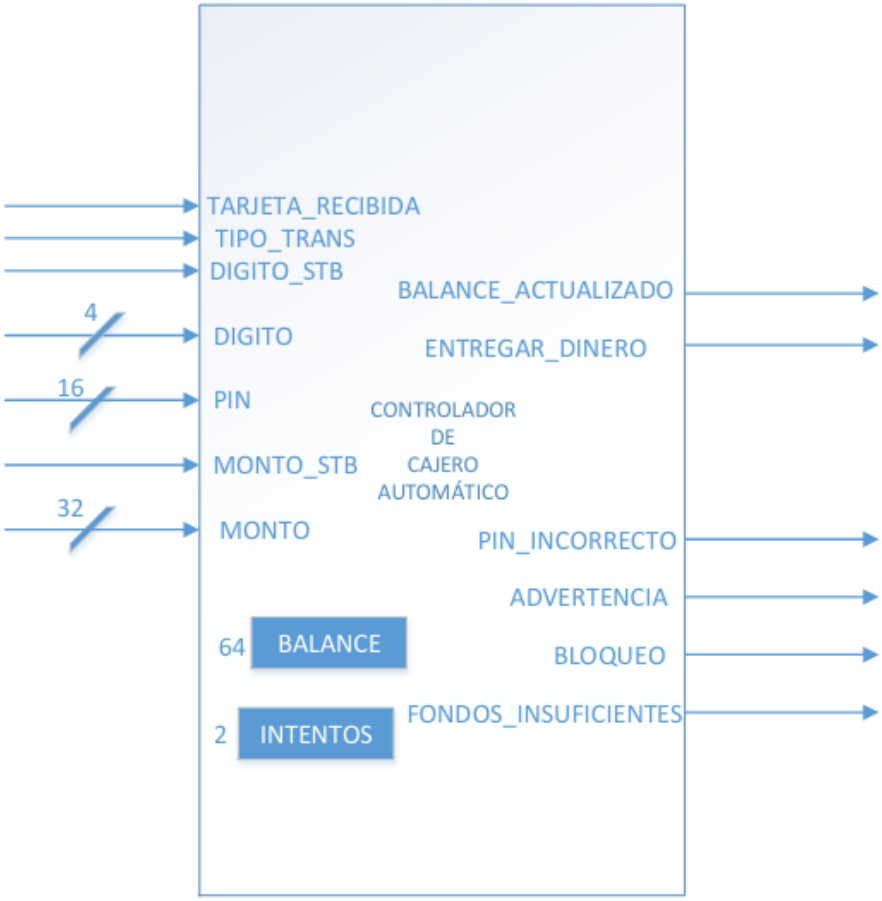

Figura 1: Diagrama de Bloques Propuesto para el Sistema

Entonces, se puede ver que las señales que **entran** son el TARJETA RECIBIDA, TIPO TRANS, DIGITO STB, DIGITO, PIN, MONTO STB y MONTO. Las senales que **salen** son BALANCE ACTUALIZADO, ENTREGAR DINERO, PIN INCORRECTO ADVERTENCIA, BLOQUEO Y FONDOS INSUFIENTES son las que salen. Ademas, se pueden ver los registros internos de BALANCE e INTENTOS.

3. Plan de Pruebas

Entonces, de acuerdo a las pruebas establecidas anteriormente:
Prueba 1: Funcionamiento ideal, deposito.

```
22  ////////////////////////////////////////////////////////////////////
23  //Prueba 1, funcionamiento ideal, deposito
24  RESET = 0;
25  TARJETA_RECIBIDA = 0;
26  PIN = 0;
27  DIGITO = 0;
28  DIGITO_STB = 0;
29  TIPO_TRANS = 0;
30  MONTO_STB = 0;
31  MONTO = 0;
32  #10;
33  RESET = 1;
34
35  RESET = 1;
36  TARJETA_RECIBIDA = 1;
37  PIN = 16'b0110011101100111;
38
39  MONTO = 32'b00000000000000100100100111110000;//150 mil
40  MONTO_STB = 1;
41  #10;
42  MONTO_STB = 0;
43  #10;
44
45  DIGITO = 4'b0110; //Primer digito, este corresponde a 6
46  DIGITO_STB = 1;
47  #10;
48  DIGITO_STB = 0;
49  #10;
50
51  DIGITO = 4'b0111; //Segundo digito, este corresponde a 7
52  DIGITO_STB = 1;
53  #10;
54  DIGITO_STB = 0;
55  #10;
56
57  DIGITO = 4'b0110; //Primer digito, este corresponde a 6
58  DIGITO_STB = 1;
59  #10;
60  DIGITO_STB = 0;
61  #10;
62  DIGITO = 4'b0111; //Segundo digito, este corresponde a 7
63  DIGITO_STB = 1;
64  #10;
65  DIGITO_STB = 0;
66  #10;
67 ////////////////////////////////////////////////////////////////////
```

Figura 2: Codigo correspondiente a la prueba 1.

Entonces, para el código que corresponde a la prueba 1, se tiene que primero se inicializan todos los estados en 0 junto al clock. Debido a que el reset esta en 0, ya que este se activa en bajo. Sucede un ciclo de tiempo donde el reset actua, y luego se inserta la tarjeta, poniendo la senal de TARJETA RECIBIDA en 1. Con esto, ahora se pone DIGITO STB en 1, para asi actualizar el valor de DIGITO con el que se quiera digitar para aproximar al pin, que en este caso se eligio ”6767”.Primero, se pide que se digite el monto, poniendo MONTO STB en 1, y digitando el valor del monto, que en este caso es de 350 mil colones.Digamos que el mismo individuo viene otro dia para hacer un retiro, entonces el balance guarda el valor de la transaccion anterior.Tambien, se deja el TIPO DE TRANS como 1, ya que se trata de un retiro, pero en este caso ideal. El primer digito es un 6, y luego se pone DIGITO STB en 0, para simbolizar que el boton se dejo de presionar. Se repite este ciclo de poner DIGITO STB en 1, poner un digito, y poner DIGITO STB en 0; se meten los digitos 7,6, y 7. Finalizado esto, como la secuencia de digitos es igual al pin, se actu-aliza el monto. Se va a ver mas a detalle esto con las ondas.

Prueba 2: Pertenece al funcionamiento donde se ingresa la contraseña equivocada menos de 3 veces.

```
////////////////////////////////////////////////////////////////
//Prueba 2, funcionamiento ideal, retiro con fondos insuficientes
  RESET = 0;
  TARJETA_RECIBIDA = 0;
  PIN = 0;
  DIGITO = 0;
  DIGITO_STB = 0;
  TIPO_TRANS = 0;
  MONTO_STB = 0;
  MONTO = 0;
  #10;
  RESET = 1;

  RESET = 1;
  TARJETA_RECIBIDA = 1;
  TIPO_TRANS = 1;
  PIN = 16'b0110011101100111;
  MONTO = 32'b00000000000001010101011100110000; //350 mil para retirar
  MONTO_STB = 1;
  #10;
  MONTO_STB = 0;
  #10;

  TARJETA_RECIBIDA = 1;
  DIGITO = 4'b0110; //Primer digito, este corresponde a 6
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;

  DIGITO = 4'b0111; //Segundo digito, este corresponde a 7
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;

  DIGITO = 4'b0110; //Tercer digito, este corresponde a 6
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;
  DIGITO = 4'b0111; //Cuarto digito, este corresponde a 7
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;
////////////////////////////////////////////////////////////////
```

Figura 3: Codigo correspondiente a la prueba 2.

Entonces, para el código que corresponde a la prueba 2, se tiene que es bastante parecido al de la prueba uno, ya que se sigue midiendo el caso ideal. Primero se inicializan todos los estados en 0 junto al clock. Debido a que el reset esta en 0, ya que este se activa en bajo+. Sucede un ciclo de tiempo donde el reset actua, y luego se inserta la tarjeta, poniendo la senal de TARJETA RECIBIDA en 1. Con esto, ahora se pone DIGITO STB en 1, para asi actualizar el valor de DIGITO con el que se quiera digitar para aproximar al pin, que en este caso se eligio "6767".Primero, se pide que se digite el monto, poniendo MONTO STB en 1, y digitando el valor del monto, que en este caso es de 150 mil colones. Tambien, se deja el TIPO DE TRANS como 0, ya que se trata de un deposito. El primer digito es un 6, y luego se pone DIGITO STB en 0, para simbolizar que el boton se dejo de presionar. Se repite este ciclo de poner DIGITO STB en 1, poner un digito, y poner DIGITO STB en 0; se meten los digitos 7,6, y 7. Finalizado esto, como la secuencia de digitos es igual al pin, se actualiza el monto. Se va a ver mas a detalle esto con las ondas.

Prueba 3: Funcionamiento normal básico, con retiro de montos insuficientes.

```
////////////////////////////////////////////////////////////////////
//Prueba 3, funcionamiento ideal, retiro con fondos suficientes
  RESET = 0;
  TARJETA_RECIBIDA = 0;
  PIN = 0;
  DIGITO = 0;
  DIGITO_STB = 0;
  TIPO_TRANS = 0;
  MONTO_STB = 0;
  MONTO = 0;
  #10;
  RESET = 1;

  RESET = 1;
  TARJETA_RECIBIDA = 1;
  TIPO_TRANS = 1;
  PIN = 16'b0110011101100111;
  MONTO = 32'b00000000000001001001001111100000; //300 mil para retirar
  MONTO_STB = 1;
  #10;
  MONTO_STB = 0;
  #10;

  TARJETA_RECIBIDA = 1;
  DIGITO = 4'b0110; //Primer digito, este corresponde a 6
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;

  DIGITO = 4'b0111; //Segundo digito, este corresponde a 7
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;

  DIGITO = 4'b0110; //Tercer digito, este corresponde a 6
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;
  DIGITO = 4'b0111; //Cuarto digito, este corresponde a 7
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;
////////////////////////////////////////////////////////////////////
```

Figura 4: Codigo correspondiente a la prueba 3.

Entonces, para el código que corresponde a la prueba 3, se tiene que primero se inicializan todos los estados en 0 junto al clock. Reiniciamos el valor anterior, se inserta la tarjeta nuevamente. EL mismo individuo viene a hacer otro retiro. Recuerda su pin, lo digita bien, se sigue la misma secuencia de MONTO, MONTO STB y DIGITO, DIGITO STB. Sin embargo, esta vez se le olvido que ya habia retirado 350 mil, entonces cuando viene a retirar 300 mil de nuevo, esta vez se dispara la alarma de fondos insufientes, y no se levanta la senal de obtener dinero. Pobrecito :(.

Prueba 4: Prueba de estrés, se levantan advertencias, bloqueo y pines incorrectos. Luego, se hace un depósito.

```
//Prueba 4, funcionamiento no ideal, se levantan alertas y pines incorrectos, bloqueos, deposito.
  RESET = 0;
  TARJETA_RECIBIDA = 0;
  PIN = 0;
  DIGITO = 0;
  DIGITO_STB = 0;
  TIPO_TRANS = 0;
  MONTO_STB = 0;
  MONTO = 0;
  #10;
  RESET = 1;

  RESET = 1;
  TARJETA_RECIBIDA = 1;
  PIN = 16'b0110011101100111;
  MONTO = 32'b00000000000000010010010011110000; //150 mil para depositar
  MONTO_STB = 1;
  #10;
  MONTO_STB = 0;
  #10;

  TARJETA_RECIBIDA = 1;
  DIGITO = 4'b0110; //Primer digito, este corresponde a 6
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;

  DIGITO = 4'b0111; //Segundo digito, este corresponde a 7
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;

  DIGITO = 4'b0001; //Tercer digito, este corresponde a 1. Como es incorrecto, se reinician los digitos.
  DIGITO_STB = 1;   //Contador de errores llega a 1.
  #10;              //Se levanta Pin Incorrecto.
  DIGITO_STB = 0;
  #10;
  DIGITO = 0;

  DIGITO = 4'b0110; //Primer digito, este corresponde a 6
  DIGITO_STB = 1;
  #10;
  DIGITO_STB = 0;
  #10;
```

Figura 5: Codigo correspondiente a la prueba 4.

Bueno, para esta prueba, se fuerzan valores incorrectos de pin, para asi levantar las senales de advertencia y pin incorrecto. Eventualmente, se llega al bloqueo, lo que obliga reiniciar la senal. Despues, cuando la contrasena es correcta, se procede a hacer el deposito. Se hizo otra prueba, donde se ve hace lo mismo para levantar todas las alarmas, pero se hace un retiro al final. Por cuestiones de espacio no se va a incluir aqui.

4. Instrucciones de Utilizacion

Para ejecutar el programa, se necesita SOLO UN COMANDO, ya entendi como se hace un makefile :DDDDD

- **make tarea:** Para obtener el sintetizado y ver las ondas de la simulacion de la implementacion conductual.
- **make tarea1:** Para simular el sintetizado. IMPORTANTE, cambiar el include en el archivo que termina en "tb", y comentar el include anterior. Por ejemplo, el include ORIGINAL es Çajero win", cuando se quiera probar el otro modulo, se debe poner 'Cajero synth'.

5. Ejemplos de la Sintesis

```
12.1.2. Re-integrating ABC results.
ABC RESULTS:               NAND cells:      634
ABC RESULTS:                NOR cells:      604
ABC RESULTS:                NOT cells:      306
ABC RESULTS:        internal signals:     1452
ABC RESULTS:           input signals:      132
ABC RESULTS:          output signals:       94
Removing temp directory.
```

Figura 6: Celdas usadas en el diseno

Es...interesante que hayan quedado tantas compuertas en la implementacion de este diseno. En lo personal, debe ser por la utilizacion de la funcion case que se uso para implementar el diseno. De ahi en fuera, el comportamiento de las ondas es el esperado, y concuerda con lo que se esperaba.

```
11. Executing DFFLIBMAP pass (mapping DFF cells to sequential cells from liberty file).
  cell DFF (noninv, pins=3, area=18.00) is a direct match for cell type $_DFF_P_.
  create mapping for $_DFF_N_ from mapping for $_DFF_P_.
  final dff cell mappings:
    DFF _DFF_N_ (.C(~C), .D( D), .Q( Q));
    DFF _DFF_P_ (.C( C), .D( D), .Q( Q));
    unmapped dff cell: $_DFF_NN0_
    unmapped dff cell: $_DFF_NN1_
    unmapped dff cell: $_DFF_NP0_
    unmapped dff cell: $_DFF_NP1_
    unmapped dff cell: $_DFF_PN0_
    unmapped dff cell: $_DFF_PN1_
    unmapped dff cell: $_DFF_PP0_
    unmapped dff cell: $_DFF_PP1_
    unmapped dff cell: $_DFFSR_NNN_
    unmapped dff cell: $_DFFSR_NNP_
    unmapped dff cell: $_DFFSR_NPN_
    unmapped dff cell: $_DFFSR_NPP_
    unmapped dff cell: $_DFFSR_PNN_
    unmapped dff cell: $_DFFSR_PNP_
    unmapped dff cell: $_DFFSR_PPN_
    unmapped dff cell: $_DFFSR_PPP_
Mapping DFF cells in module `\Cajero':
```

Figura 7: Flops utilizados en el diseno

Tambien se pueden notar, y, lastimosamente, como quedo una cantidad tan grande de latches, no fue posible comprobar el funcionamiento del circuito sintetizado y compararlo con las ondas del circuito original. Esto nos lleva a concluir que, con esta implementacion el circuito no es sintetizable, o bien, hay errores que se pueden corregir, pero se salen de lo aprendido hasta el momento. Con conocimiento futuro bien puede ser posoble lograr esta implementacion.

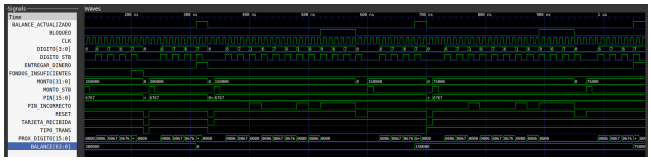

Figura 8: Comportamiento General de las ondas del circuito sin sintetizar

6. Conclusiones

Se utilizaron aproximadamente 2 flops para desarrollar el controlador. Sin embargo, potencialmente se podia llegar hasta 10 u 11. Se puede ver, tambien, que quedaron unos 80 latches, que si me pregunta a mi? Ni idea como llegaron ahi. Se que la teoria para que no salga un latch es que hay que cubrir todos los casos posibles para que el programa de Yosys no tenga que inferir nada, pero es solo una teoria. La prescencia tan grande de latches en el circuito sintetizado es por el uso de cases en cadenas de 32 bits, ademas de que se utilizaron 4 cases? No tengo ni idea.