

Máquina de estados finitos (FSM)

Proyecto Cajero Automático

I. Descripción del proyecto:

Este proyecto consiste en una máquina de estados finitos que realiza el funcionamiento de un cajero automático personalizado, el cual tiene una opción de encendido y apagado, junto con una entrada de múltiples bits, para ingresar la contraseña predeterminada del cajero para poder loggarse y tener acceso las opciones de consultar el saldo, o bien retirar dinero en montos de Q.100.00, con un límite total de retiro de Q.700.00, control el cual es llevado por un contador de 3 bits, lo cual se puede escalar en futuras versiones de la máquina para retirar en montos mayores y con un límite mayor.

Para mostrar las salidas del cajero se utilizan leds, los cuales muestran es estado ON/OFF de la máquina, si esta loggeado, si se esta retirando si se ha llegado al limite de retiro, y displays los cuales muestran los valores de los dígitos del pin ingresado, así como también para mostrar en pantalla el valor de del contador cuando se consulta. La máquina en con junto esta conformada por 4 FSM factorizas, 1 de ellas reutilizada 3 veces, las cuales realizan una acción determinada para que el cajero funcione correctamente, tal y como se observa en la figura 1.

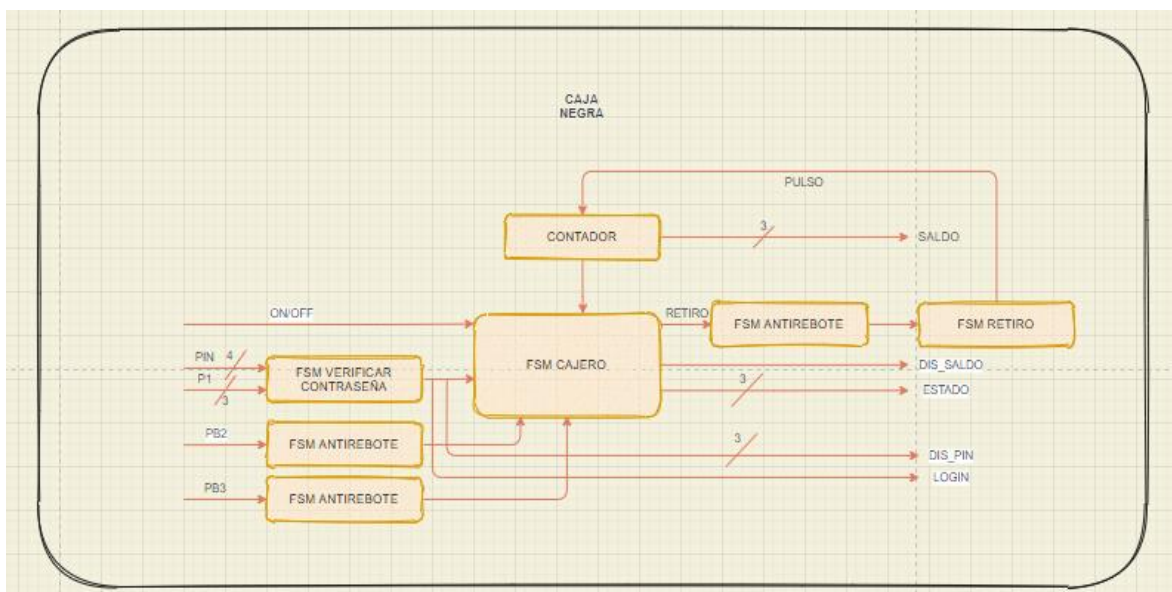


Figura 1. Caja negra de la FSM Cajero Automático

II. FSM Antirebote:

La máquina anti-rebote es una FSM, la cual tiene el funcionamiento de recibir un pulso de entrada de x cantidad de tiempo, como lo puede ser un push button, lo cual al tener nuestro clock períodos de reloj de milisegundos, se puede interpretar como que el push fue pulsado no una sino varias veces, por ello que se implementa el anti-rebote el cual evita este problema, enviando solo un pulso cada vez que se presiona el push button.

Como se observa en la figura 2, se puede ver los diagramas de estados tanto como una FSM de Mealy, como una de Moore. En este proyecto se implemento como una FSM de Mealy ya que era mucho más sencilla. Como se observa, se realizo el antirebote, pero se reutilizó para más de una entrada, para evitar el efecto de rebote en nuestro circuito de la máquina cajero.

El funcionamiento de esta máquina, consiste en que cuando la señal de entrada es un cero lógico, esta se mantiene en estado S0 con una salida de cero lógico, pero al momento que la entrada cambia a 1, como salida tenemos un 1 lógico y esta cambia al estado S1, en donde mientras la señal siga siendo 1, la salida es cero, y si se deja de tener la señal de entrada en 1, esta se regresa a su estado inicial S0.

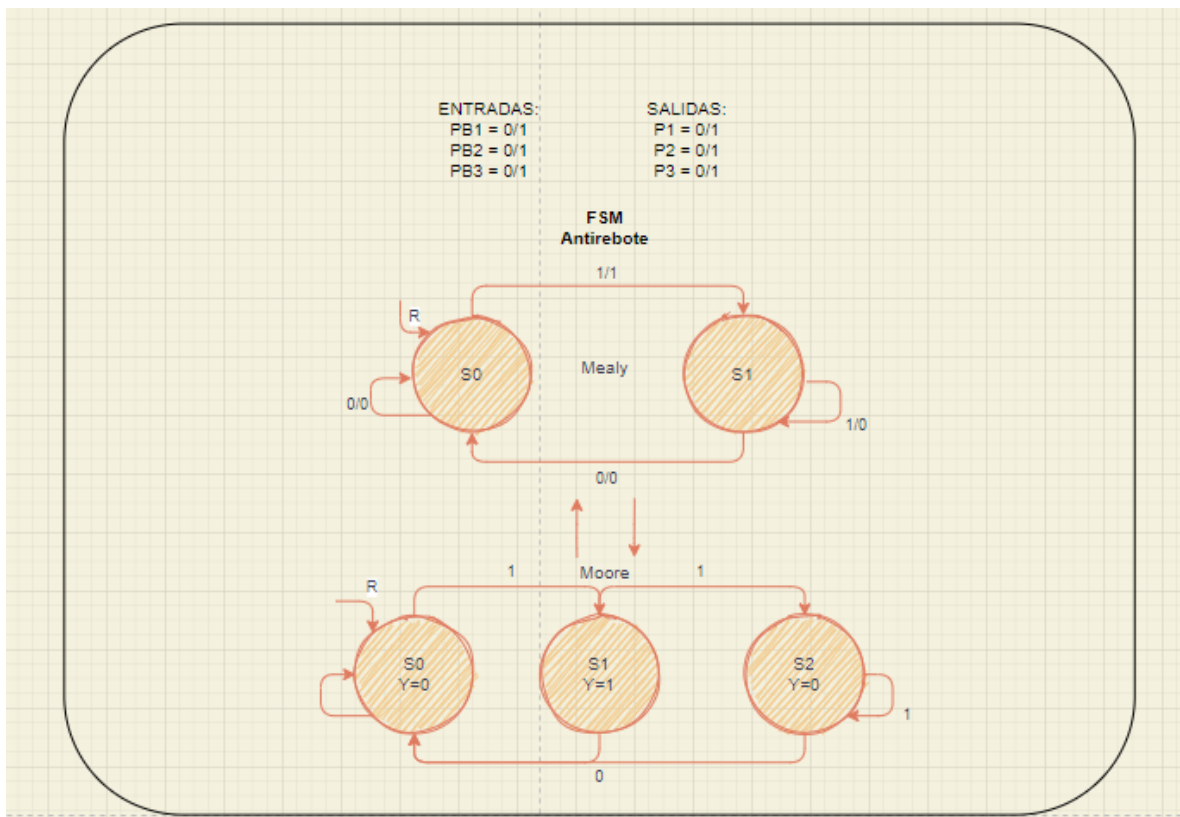


Figura 2. Diagrama de estados de la FSM antirebote

Tabla de transición de estados y salidas (mealy)			
Current State	Input	Next State	Output
S0	0	S0	0
S0	1	S1	1
S1	0	S0	0
S1	1	S1	0

Figura 3. Tabla de estados y salidas de la FSM anti-rebote sin codificar

Tabla de transición de estados y salidas con codificación binaria			
Current State	Input	Next State	Output
0	0	0	0
0	1	1	1
1	0	0	0
1	1	1	0

Figura 4. Tabla de estados y salidas de la FSM anti-rebote codificada

III. FSM Retiro:

Para la máquina de retiro de Q.100.00, como se observa en la figura 5, se implementó como una FSM de moore, la cual mientras no recibe ninguna señal se mantiene en el estado S0, y una salida de Pulso= 0, al recibir una señal de entrada que indica que se quiere retirar Q.100.00, entonces pasa del estado S0, al estado S1, en donde Pulso=1, y seguidamente sin necesidad de alguna otra señal, pasa el estado S2 en donde P=0 para finalmente regresar al estado S0.

Como se puede intuir, la señal de salida de la FSM, de retiro es una señal cuadrada con un ciclo positivo, esto ya que dentro del funcionamiento del proyecto se sustituyó la señal de clock en el contador implementado por el Pulso de salida de esta máquina, para que cada vez que se realizara un retiro dicha señal que llegara al contador de 3 bits y así variar este en 1 unidad, para retirar Q.100.00 de lo que haya en existencia.

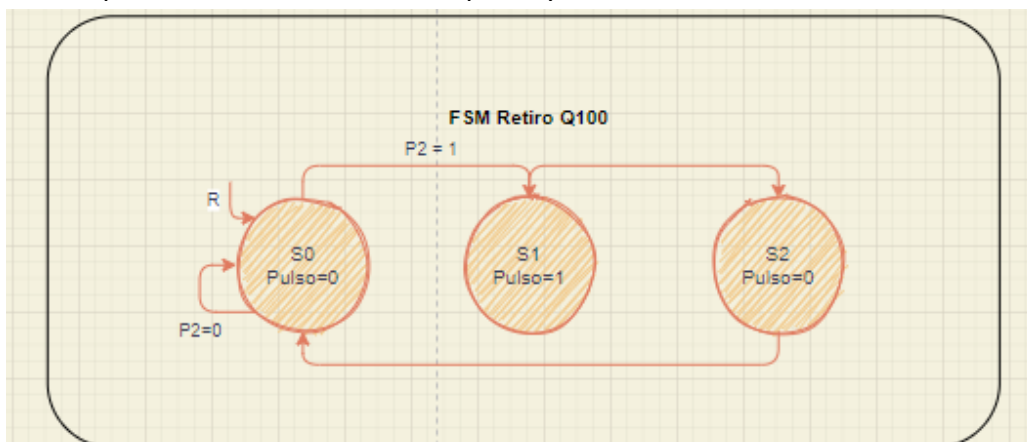


Figura 5. Diagrama de estados de la FSM de retiro

Tabla de transición de estados		
Current State	INPUT	Next State
S0	0	S0
S0	1	S1
S1	X	S2
S2	X	S0

Figura 6. Tabla de estados de la FSM retiro sin codificar

Tabla de salidas	
State	Y
S0	0
S1	1
S2	0

Figura 7. Tabla de salidas de la FSM retiro sin codificar

Tabla de transición de estados con codificación binaria				
Current State		INPUT	Next State	
S1	S0	PS	S1'	S0'
0	0	0	0	0
0	0	1	0	1
0	1	X	1	0
1	0	X	0	0

Figura 8. Tabla de estados de la FSM retiro codificada

Tabla de salidas codificada		
State		OUTPUT
S1	S0	Y
0	0	0
0	1	1
1	0	0

Figura 9. Tabla de salidas de la FSM retiro codificada

IV. FSM Verificación

Esta máquina de estado finitos es el sistema de login del cajero, la cual tiene una contraseña predefinida 269. Dicha máquina como se puede observar en la figura 10, recibe 2 entradas de múltiples bits, un Pin de 4 bits, el cual representa la entrada de un número base 10, y una entrada P, que representa el interruptor por cada dígito del pin.

Inicialmente, la máquina no recibe ninguna señal y se mantiene en el estado S0, posteriormente se ingresa el primer pin y se prende el primer interruptor es decir esta señal es P=001, la máquina verifica Pin=0010, es decir un dos en base 10, de lo contrario se

mantiene en el estado S0 con salidas Dis_pin=000 y Login=0, si es correcto entonces se pasa al estado S1 en donde Login=0 nuevamente, pero se prende el primer display con una salida Dis_pin =001. Luego, se ingresa el segundo pin y si Pin=0110 (seis en base 10), y P=011, entonces pasa al estado S2 donde la salida Login=0 y Dis_pin=011. Se verifica el último dígito del pin y mientras P!=1001 (nueve en base 10), o bien P!=111, se mantiene en el estado S2, pero cuando ambos datos de entrada concuerdan pasa al estado S3 en donde la señal de Dis_pin = 111, y Login = 1, lo cual significa que se ha loggeado correctamente.

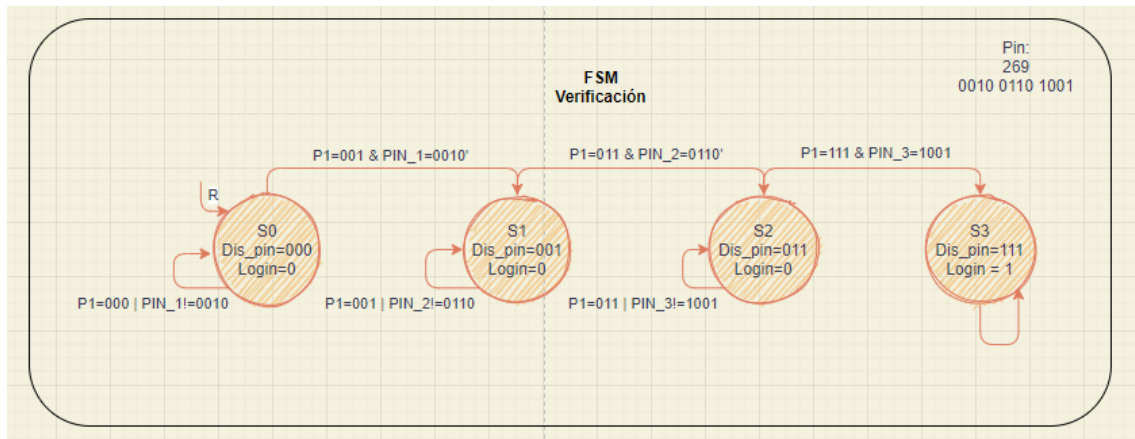


Figura 10. Diagrama de estados de la FSM verificación

Tabla de transición de estados								
Current State	P3	P2	P1	D	C	B	A	Next State
S0	0	0	0	X	X	X	X	S0
S0	0	0	1	0	0	1	0	S1
S1	0	0	1	X	X	X	X	S1
S1	0	1	1	0	1	1	0	S2
S2	0	1	1	X	X	X	X	S2
S2	1	1	1	1	0	0	1	S3
S3	X	X	X	X	X	X	X	S3

Figura 11. Tabla de transición de estados de la FSM verificación sin codificar

Tabla de salidas				
State	Login	Disp3	Disp2	Disp1
S0	0	0	0	0
S1	0	0	0	1
S2	0	0	1	1
S3	1	1	1	1

Figura 12. Tabla de salidas de la FSM verificación sin codificar

Tabla de transición de estados con codificación binaria										
Current State		INPUTS							Next State	
S1	S0	P3	P2	P1	D	C	B	A	S1'	S0'
0	0	0	0	0	X	X	X	X	0	0
0	0	0	0	1	0	0	1	0	0	1
0	1	0	0	1	X	X	X	X	0	1
0	1	0	1	1	0	1	1	0	1	0
1	0	0	1	1	X	X	X	X	1	0
1	0	1	1	1	1	0	0	1	1	1
1	1	X	X	X	X	X	X	X	1	1

Figura 13. Tabla de transición de estados de la FSM verificación codificada

Tabla de salidas codificada					
State		OUTPUTS			
S1	S0	Login	Disp3	Disp2	Disp1
0	0	0	0	0	0
0	1	0	0	0	1
1	0	0	0	1	1
1	1	1	1	1	1

Figura 14. Tabla de salidas de la FSM verificación codificada

V. FSM Cajero

La máquina de cajero tiene el objetivo de manejar el conjunto de las demás máquinas descritas anteriormente, junto con el contador el cual es un elemento, que lleva el registro de dinero que se tiene en el cajero y que guarda la memoria de este aun cuando el cajero se apaga.

Como se observa en la figura 15, la máquina se mantiene en estado S0, mientras la entrada del interruptor ON sea un cero lógico, o bien mientras ya no haya dinero en existencia. Cuando el cajero se enciende este pasa a un estado encendido, en el cual se espera a que se realice el login en la FSM de verificación y cuando esta tenga una salida de Login = 1, esta señal ingresa al cajero, el cual pasa al estado S2, en el cual es el Home del cajero en donde se mantiene hasta que el usuario realice una acción ya sea de retirar, en cuyo caso pasa al estado S3, o bien consultar en donde pasa al estado S4 y se tiene una salida de Dis_saldo=1, en donde se enciende el display del contador.

En cualquiera de los dos casos, la FSM verifica que el interruptor de encendido esté en 1 y que el contador siga con un valor mayor a cero, de ser así, la máquina vuelve al estado S2, en donde espera alguna otra instrucción y se mantiene hasta que se retire, consulte o bien se apague la máquina de cajero.

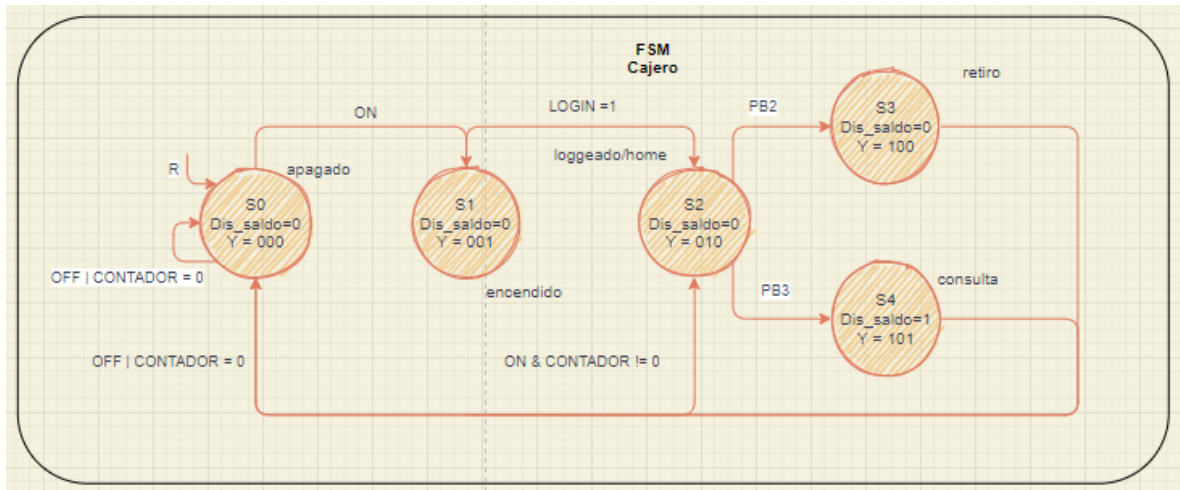


Figura 15. Diagrama de estados de la FSM Cajero

Tabla de transición de estados						
Current State	ON	CONT.	LOGIN	PB2	PB1	Next State
S0	0	X	X	X	X	S0
S0	X	0	X	X	X	S0
S0	1	1	X	X	X	S1
S1	0	X	X	X	X	S0
S1	1	1	0	X	X	S1
S1	1	1	1	X	X	S2
S2	X	X	1	1	0	S3
S2	X	X	1	0	1	S4
S3	1	1	1	X	X	S1
S3	0	X	X	X	X	S0
S3	X	0	X	X	X	S0
S4	1	1	1	X	X	S1
S4	0	X	X	X	X	S0
S4	X	0	X	X	X	S0

Figura 16. Tabla de transición de estados de la FSM Cajero sin codificar

Tabla de salidas				
State	Y2	Y1	Y0	DISP_SALDO
S0	0	0	0	0
S1	0	0	1	0
S2	0	1	0	0
S3	1	0	0	0
S4	1	0	1	1

Figura 17. Tabla de salidas de la FSM Cajero sin codificar

Tabla de transición de estados con codificación binaria										
Current State			INPUTS					Next State		
S2	S1	S0	ON	CONT.	LOGIN	PB2	PB1	S2'	S1'	S0'
0	0	0	0	X	X	X	X	0	0	0
0	0	0	X	0	X	X	X	0	0	0
0	0	0	1	1	X	X	X	0	0	1
0	0	1	0	X	X	X	X	0	0	0
0	0	1	1	1	0	X	X	0	0	1
0	0	1	1	1	1	X	X	0	1	0
0	1	0	X	X	1	1	0	0	1	1
0	1	0	X	X	1	0	1	1	0	0
0	1	1	1	1	1	X	X	0	0	1
0	1	1	0	X	X	X	X	0	0	0
0	1	1	X	0	X	X	X	0	0	0
1	0	0	1	1	1	X	X	0	0	1
1	0	0	0	X	X	X	X	0	0	0
1	0	0	X	0	X	X	X	0	0	0

Figura 18. Tabla de transición de estados de la FSM Cajero codificada

Tabla de salidas codificada						
State			OUTPUTS			
S2	S1	S0	Y2	Y1	Y0	DISP_SALDO
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
1	0	0	0	1	0	1

Figura 19. Tabla de salidas de la FSM Cajero codificada

VI. Logic Friday y ecuaciones booleanas

Se ingresó las tablas de transición de estados y de salidas codificadas de cada una de las FSM en el software logic Friday y se generó las ecuaciones booleanas en cada uno de los casos:

Functi...	Inputs	Outputs	True	False	DC	PI	Gates
F1-F0	9	2	257, ...	255, ...	0, 0	12	Not mapped

S1	S0	P3	P2	P1	D	C	B	A	=>	F1	F0
X	1	0	1	1	0	1	1	0		1	
0	X	0	0	1	0	0	1	0		1	
1	X	1	1	1	1	0	0	1		1	
1	X	X	X	X	X	X	X	X		1	
X	1	X	X	0	X	X	X	X		1	
X	1	X	X	X	X	X	X	1		1	
X	1	X	X	X	X	0	X	X		1	
X	1	X	X	1	X	X	X	X		1	
X	1	X	0	X	X	X	X	X		1	
X	1	1	X	X	X	X	X	X		1	
X	1	X	X	X	0	X	X	X		1	
1	1	X	X	X	X	X	X	X		1	

Minimized:

F1 = S0 P3' P2 P1 D' C B A' + S1 ;

F0 = S1' P3' P2' P1 D' C' B A' + S1 P3 P2 P1 D C' B' A' + S0 P1' + S0 A + S0 B' + S0 D + S0 P2' + S0 P3 + S0 C' + S1 S0 ;

Figura 20. Ecuaciones booleanas FSM Verificación (estados)

Funci...	Inputs	Outputs	True	False	DC	PI	Gates	
F1-F0	9	2	257, ...	255, ...	0, 0	12	Not mapped	
DIS3-L...	2	4	1, 2, ...	3, 2, ...	0, 0, ...	3	Not mapped	

S1	S0	=>	DIS3	DIS2	DIS1	LOGIN	
1	1		1			1	Entered by truthtable:
X	1				1		DIS3 = S1 S0;
1	X			1	1		DIS2 = S1 S0' + S1 S0;
							DIS1 = S1' S0 + S1 S0' + S1 S0;
							LOGIN = S1 S0;
							Minimized:
							DIS3 = S1 S0;
							DIS2 = S1 ;
							DIS1 = S0 + S1 ;
							LOGIN = S1 S0;

Figura 21. Ecuaciones booleanas FSM Verificación (salidas)

Funci...	Inputs	Outputs	True	False	DC	PI	Gates	
DIS3-L...	2	4	1, 2, ...	3, 2, ...	0, 0, ...	3	Not mapped	
N.state-Y	2	2	2, 1	2, 3	0, 0	2	Not mapped	

A.state	PB	=>	N.state	Y	
X	1		1		Entered by truthtable:
0	1			1	N.state = A.state' PB + A.state PB;
					Y = A.state' PB;
					Minimized:
					N.state = PB;
					Y = A.state' PB;
					Minimized:
					N.state = PB;
					Y = A.state' PB;

Figura 22. Ecuaciones booleanas FSM Anti-rebote

Funci...	Inputs	Outputs	True	False	DC	PI	Gates	
N.state-Y	2	2	2, 1	2, 3	0, 0	2	Not mapped	
F1-F0(2)	3	2	2, 1	4, 5	2, 2	2	Not mapped	

S1	S0	P2	=>	F1	F0	
X	1	X		1		Entered by truthtable:
0	0	1			1	F1 = S1' S0 P2' + S1' S0 P2;
						F0 = S1' S0' P2;
						Minimized:
						F1 = S0 ;
						F0 = S1' S0' P2;

Figura 23. Ecuaciones booleanas FSM Retiro (estados)

Funci...	Inputs	Outputs	True	False	DC	PI	Gates	
F1-F0(2)	3	2	2, 1	4, 5	2, 2	2	Not mapped	
PULSO	2	1	1	2	1	1	Not mapped	

S1	S0	=>	PULSO	
X	1		1	Entered by truthtable:
				PULSO = S1' S0;
				Minimized:
				PULSO = S0;

Figura 24. Ecuaciones booleanas FSM Retiro (salida)

Funci...	Inputs	Outputs	True	False	DC	PI	Gates
PULSO	2	1	1	2	1	1	Not mapped
F2-F0	8	3	2, 10, ...	158, ...	96, 9...	10	Not mapped

S2	S1	S0	ON	CONT	LOGIN	PB2	PB1	=>	F2	F1	F0
X	1	1	1	1	X	X	X				1
X	0	0	1	1	X	X	X				1
X	0	1	1	0	X	X	X				1
X	0	1	1	1	1	X	X			1	
X	1	0	1	1	X	0	X			1	
X	1	0	1	1	X	X	1			1	
X	0	1	X	0	0	0	0				1
X	1	0	1	1	X	1	0		1		
X	X	1	1	1	0	X	X				1
X	X	0	1	1	X	0	1				1

Minimized:

F2 = S1 S0' ON CONT PB2 PB1';

F1 = S1' S0 ON CONT LOGIN + S1 S0' ON CONT PB2' + S1 S0' ON CONT PB1;

F0 = S1 S0 ON CONT + S1' S0' ON CONT + S1' S0 ON CONT' + S1' S0 CONT' LOGIN' PB2' PB1' + S0 ON CONT LOGIN' + S0' ON CONT PB2' PB1;

Figura 25. Ecuaciones booleanas FSM Cajero (estados)

VII. Diagrama de Timing

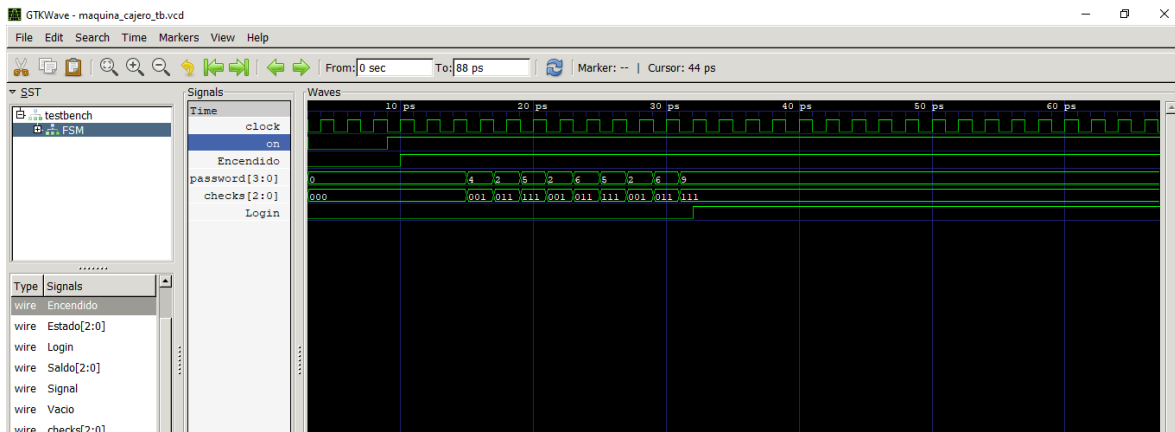


Figura 27. Prueba con diagrama de timing GTKWave del login y cajero

Seguidamente se comprobó el funcionamiento interno de cajero con las opciones de consulta, la opción de retiro y el anti-rebote. Como se observa en la figura 28, al presionar los push de retiro y de consulta la señal se genera por más de un ciclo de reloj, sin embargo, la implementación del anti-rebote, permite que la señal de retiro y consulta que entra a cajero sea un solo pulso que se vuelve cero nuevamente, sin importar que el push este presionado por más de un ciclo, el cual es el funcionamiento esperado por el antirebote.

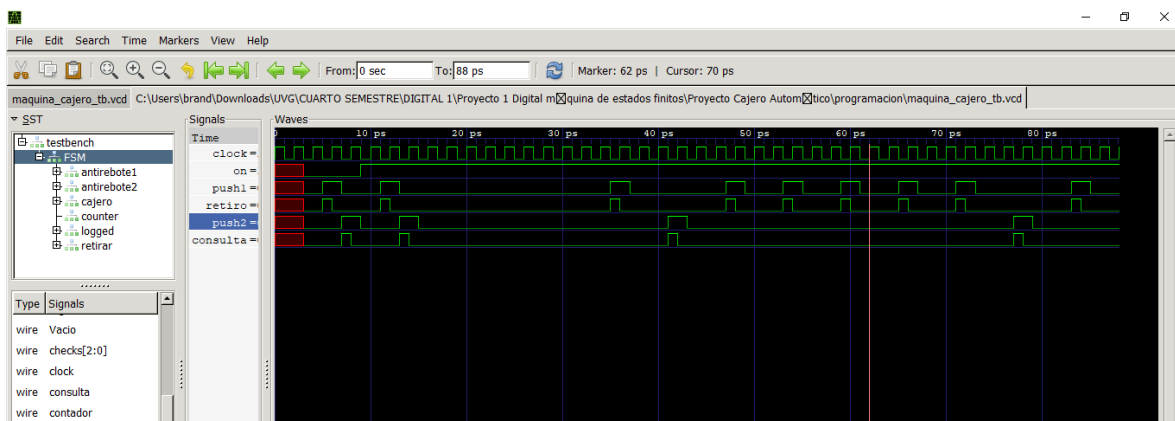


Figura 28. Prueba con diagrama de timing GTKWave de antirebote, consulta y retiro

Finalmente se procedió a comprobar el funcionamiento correcto, de la máquina de cajero completa mediante el diagrama de timing, como se observa en la figura 29. Como se nota, el funcionamiento del antirebote y el login es correcto, por otro lado, las señales de retiro y consulta solo generan un cambio en Signal y en Display_Saldo cuando la máquina esta en funcionamiento, es decir cuando esta encendida y loggeado ya que cuando no lo este, no debe existir cambio, aunque las señales del consulta y retiro estén encendidas. También se puede notar el cambio de valor del contador cada vez que se realiza un retiro después del respectivo tiempo de espera, que se implementó como un delay que representa el tiempo durante el cual se muestra el saldo y se realiza de retiro de los Q.100.00.

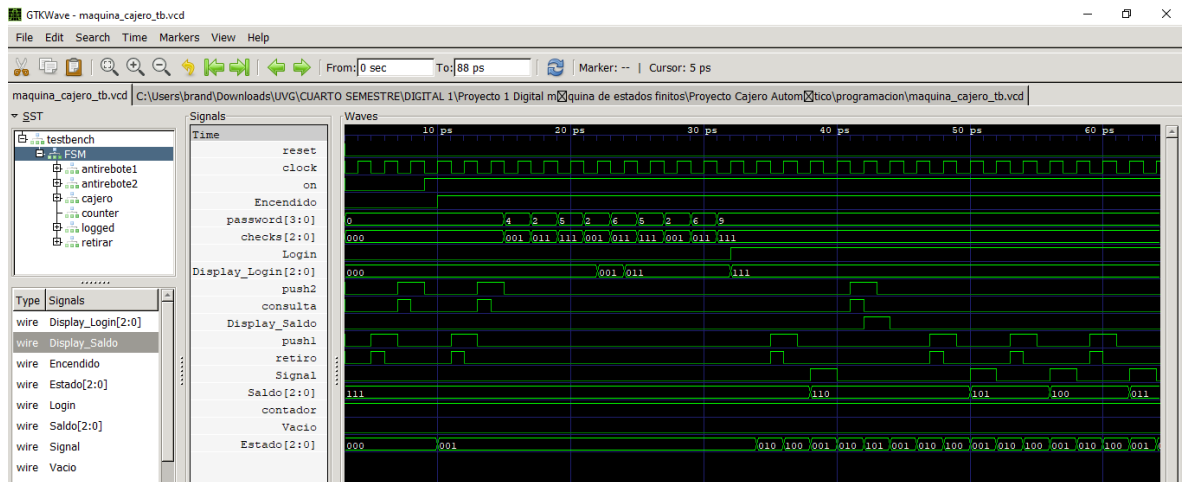


Figura 29. Prueba con diagrama de timing GTKWave de la máquina completa con contador

VIII. Implementación en verilog

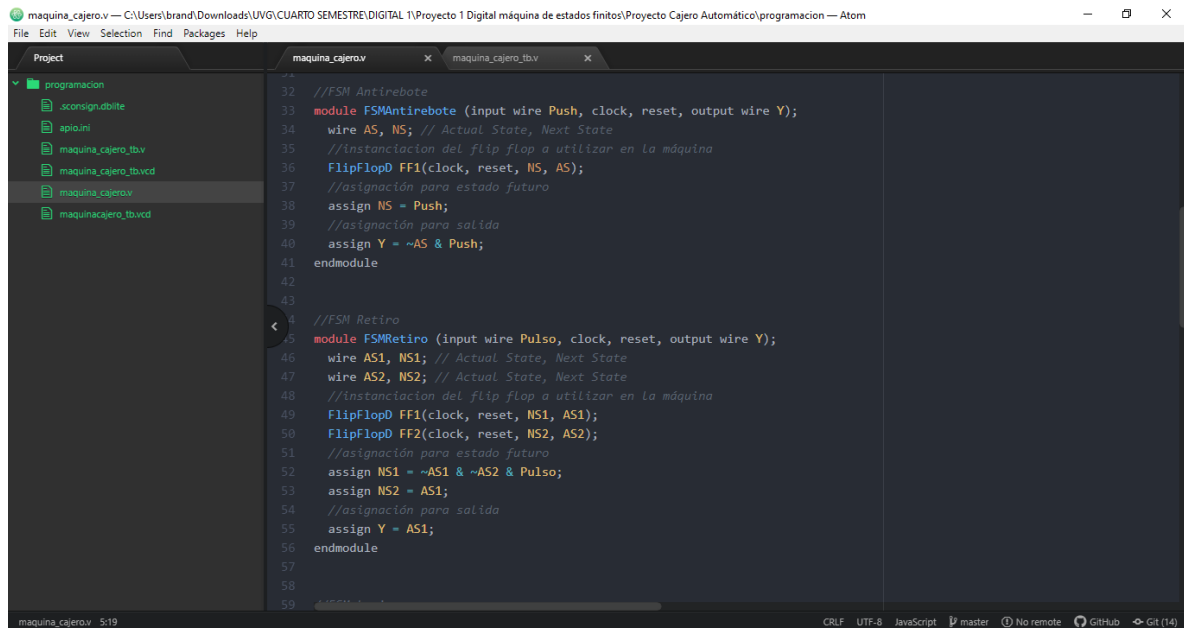
Se implemento cada una de las máquinas y la máquina completa del cajero automático en system verilog implementando programación behavioral con las ecuaciones booleanas y creando los distintos módulos tanto de los FlipFlop tipo D utilizados y contador junto con su testbench, mostrando en consola cada una de las posibles combinaciones necesarias.

```

Project
├── programacion
│   ├── .sconsignidbfile
│   ├── aplo.ini
│   ├── maquina_cajero_tbv
│   ├── maquina_cajero_tbvcd
│   ├── maquina_cajero.v
│   └── maquina_cajero_tbvcd
└── maquina_cajero.v
    1 //Brandon Amisael Garrido Ramirez
    2 //Carnet 19421
    3 //Proyecto Cajero Automático
    4
    5 //Flip flop tipo D
    6 module FlipFlopD(input wire clock, reset, input wire D, output reg Y);
    7     always @ (posedge clock, posedge reset) begin
    8         if (reset) begin
    9             Y <= 1'b0;
    10        end
    11        else begin
    12            Y <= D;
    13        end
    14    end
    15 endmodule
    16
    17 // contador inverso desde 111 base 2, 7 en base 10 hasta cero
    18 // La entrada retiro, sustituye la señal del clock en el contador
    19 module Down_Counter(input wire retiro, reset, output reg[2:0] count);
    20     initial count = 3'b111;
    21     always @ (posedge retiro or posedge reset) begin
    22         if(reset) begin
    23             count <= 3'b111;
    24         end
    25         else begin
    26             count <= count - 3'd1;
    27         end
    28     end
    29 endmodule

```

Figura 30. Código en verilog 1



The screenshot shows a Verilog code editor with two tabs: 'maquina_cajero.v' and 'maquina_cajero_tb.v'. The 'maquina_cajero.v' tab is active, displaying the code for two modules. The first module, 'FSMAntirebote', is defined with inputs 'Push', 'clock', 'reset', and 'output wire Y'. It uses a 'FlipFlopD' component 'FF1' and assigns 'NS' to 'Push' and 'Y' to '~AS & Push'. The second module, 'FSMRetiro', is defined with inputs 'Pulso', 'clock', 'reset', and 'output wire Y'. It uses two 'FlipFlopD' components 'FF1' and 'FF2', and assigns 'NS1' to '~AS1 & ~AS2 & Pulso', 'NS2' to 'AS1', and 'Y' to 'AS1'. The code is written in a dark-themed editor with syntax highlighting.

```
32 //FSM Antirebote
33 module FSMAntirebote (input wire Push, clock, reset, output wire Y);
34   wire AS, NS; // Actual State, Next State
35   //instanciacion del flip flop a utilizar en la máquina
36   FlipFlopD FF1(clock, reset, NS, AS);
37   //asignación para estado futuro
38   assign NS = Push;
39   //asignación para salida
40   assign Y = ~AS & Push;
41 endmodule
42
43
44
45 //FSM Retiro
46 module FSMRetiro (input wire Pulso, clock, reset, output wire Y);
47   wire AS1, NS1; // Actual State, Next State
48   wire AS2, NS2; // Actual State, Next State
49   //instanciacion del flip flop a utilizar en la máquina
50   FlipFlopD FF1(clock, reset, NS1, AS1);
51   FlipFlopD FF2(clock, reset, NS2, AS2);
52   //asignación para estado futuro
53   assign NS1 = ~AS1 & ~AS2 & Pulso;
54   assign NS2 = AS1;
55   //asignación para salida
56   assign Y = AS1;
57 endmodule
58
59
```

Figura 31. Código en verilog 2



The screenshot shows a Verilog code editor with two tabs: 'maquina_cajero.v' and 'maquina_cajero_tb.v'. The 'maquina_cajero.v' tab is active, displaying the code for two modules. The first module, 'FSMLogin', is defined with inputs 'Pin' [3:0], 'Int' [2:0], 'clock', 'reset', and outputs 'Display' [2:0] and 'Login'. It uses two 'FlipFlopD' components 'FF1' and 'FF2', and assigns 'NS1' to '(~AS2 & ~Int[2] & ~Int[1] & Int[0] & ~Pin[3] & ~Pin[2] & Pin[1] & ~Pin[0]) | (AS2 & Int[2] & Int[1] & Int[0] & ~Pin[3] & Pin[2] & Pin[1] & ~Pin[0]) | AS2', 'NS2' to 'AS1 & ~Int[2] & Int[1] & Int[0] & ~Pin[3] & Pin[2] & Pin[1] & ~Pin[0]', 'Display[0]' to 'AS2 | AS1', 'Display[1]' to 'AS2', 'Display[2]' to 'AS2 & AS1', and 'Login' to 'AS2 & AS1'. The second module, 'FSMCajero', is defined with inputs 'on', 'contador', 'login', 'retiro', 'consulta', 'clock', 'reset', and outputs 'Estado' [2:0] and 'Di'. It uses three 'FlipFlopD' components 'FF1', 'FF2', and 'FF3', and assigns 'NS1' to 'on', 'NS2' to 'contador', and 'NS3' to 'login'. The code is written in a dark-themed editor with syntax highlighting.

```
60 module FSMLogin (input wire [3:0] Pin, input wire [2:0] Int, input wire clock, reset, output wire [2:0] Display, output wire Login);
61   wire AS1, NS1; // Actual State, Next State
62   wire AS2, NS2; // Actual State, Next State
63   //instanciacion del flip flop a utilizar en la máquina
64   FlipFlopD FF1(clock, reset, NS1, AS1);
65   FlipFlopD FF2(clock, reset, NS2, AS2);
66   //asignación para estado futuro
67
68   assign NS1 = (~AS2 & ~Int[2] & ~Int[1] & Int[0] & ~Pin[3] & ~Pin[2] & Pin[1] & ~Pin[0]) | (AS2 & Int[2] & Int[1] & Int[0] & ~Pin[3] & Pin[2] & Pin[1] & ~Pin[0]) | AS2;
69   assign NS2 = (AS1 & ~Int[2] & Int[1] & Int[0] & ~Pin[3] & Pin[2] & Pin[1] & ~Pin[0]) | AS2;
70   //asignación para salidas
71   assign Display[0] = AS2 | AS1;
72   assign Display[1] = AS2;
73   assign Display[2] = AS2 & AS1;
74   assign Login = AS2 & AS1;
75 endmodule
76
77
78 //FSM Cajero
79 module FSMCajero (input wire on, contador, login, retiro, consulta, clock, reset, output wire [2:0] Estado, output wire Di);
80   wire AS1, NS1; // Actual State, Next State
81   wire AS2, NS2; // Actual State, Next State
82   wire AS3, NS3; // Actual State, Next State
83   //instanciacion del flip flop a utilizar en la máquina
84   FlipFlopD FF1(clock, reset, NS1, AS1);
85   FlipFlopD FF2(clock, reset, NS2, AS2);
86   FlipFlopD FF3(clock, reset, NS3, AS3);
87   //asignación para estado futuro
```

Figura 32. Código en verilog 3

```

//
//FSM Cajero
78
79 module FSMCajero (input wire on, contador, login, retiro, consulta, clock, reset, output wire [2:0] Estado, output wire Display_Saldo);
80   wire AS1, NS1; // Actual State, Next State
81   wire AS2, NS2; // Actual State, Next State
82   wire AS3, NS3; // Actual State, Next State
83   //Instanciacion del flip flop a utilizar en la máquina
84   FlipFlopD FF1(clock, reset, NS1, AS1);
85   FlipFlopD FF2(clock, reset, NS2, AS2);
86   FlipFlopD FF3(clock, reset, NS3, AS3);
87   //asignación para estado futuro
88
89   assign NS1 = (AS2 & AS1 & on & contador) | (~AS2 & ~AS1 & on & contador) | (~AS2 & AS1 & on & ~contador) | (~AS2 & AS1 & ~on & contador);
90   assign NS2 = (~AS2 & AS1 & on & contador & login) | (AS2 & ~AS1 & on & contador & ~consulta) | (AS2 & ~AS1 & on & contador & consulta & ~retiro);
91   assign NS3 = AS2 & ~AS1 & on & contador & consulta & ~retiro;
92   //asignación para salidas
93   assign Estado[0] = AS3 | (AS1 & ~AS2);
94   assign Estado[1] = ~AS1 & AS2;
95   assign Estado[2] = AS3 | (AS1 & AS2);
96   assign Display_Saldo = AS3;
97 endmodule
98
99
100 //Union de todas las FSM, Máquina completa cajero automático
101 module FSMCajeroAutomatico(input wire clock, reset, on, push1, push2, input wire [3:0] password, input wire [2:0] checks,
102   output wire Login, Encendido, pulso, Display_Saldo, Vacio, output wire [2:0] Display_Login, output wire [2:0] Saldo);
103   wire contador, rese, retiro, consulta, Signal;
104   wire [2:0] Estado;
105
106   FSMCajero cajero(on, contador, Login, retiro, consulta, clock, reset, Estado, Display_Saldo);
107   FSMLogin logged(password, checks, clock, reset, Display_Login, Login);
108   FSMRetiro retirar(pulso, clock, reset, Signal);
109   Down_Counter counter(Signal, reset, Saldo);
110   FSMAntirebote antirebote1(push1, clock, reset, retiro);
111   FSMAntirebote antirebote2(push2, clock, reset, consulta);
112 endmodule

```

Figura 33. Código en verilog 4

```

93   assign Estado[0] = AS3 | (AS1 & ~AS2);
94   assign Estado[1] = ~AS1 & AS2;
95   assign Estado[2] = AS3 | (AS1 & AS2);
96   assign Display_Saldo = AS3;
97 endmodule
98
99
100 //Union de todas las FSM, Máquina completa cajero automático
101 module FSMCajeroAutomatico(input wire clock, reset, on, push1, push2, input wire [3:0] password, input wire [2:0] checks,
102   output wire Login, Encendido, pulso, Display_Saldo, Vacio, output wire [2:0] Display_Login, output wire [2:0] Saldo);
103   wire contador, rese, retiro, consulta, Signal;
104   wire [2:0] Estado;
105
106   FSMCajero cajero(on, contador, Login, retiro, consulta, clock, reset, Estado, Display_Saldo);
107   FSMLogin logged(password, checks, clock, reset, Display_Login, Login);
108   FSMRetiro retirar(pulso, clock, reset, Signal);
109   Down_Counter counter(Signal, reset, Saldo);
110   FSMAntirebote antirebote1(push1, clock, reset, retiro);
111   FSMAntirebote antirebote2(push2, clock, reset, consulta);
112 endmodule

```

Figura 34. Código en verilog 5

```

1 module testbench();
2 reg clock, reset, on, push1, push2;
3 reg [3:0] password;
4 reg [2:0] checks;
5 wire Login, Encendido, pulso, Display_Saldo, Vacio;
6 wire [2:0] Display_Login,Saldo;
7
8 always
9 begin
10     clock <= 1;
11     #1 clock <= ~clock; // se realiza el cambio del reloj
12     #1;
13 end
14
15 FSMCajeroAutomatico FSM(clock, reset, on, push1, push2, password, checks,
16     Login, Encendido, pulso, Display_Saldo, Vacio, Display_Login,Saldo);
17
18
19 initial begin
20     $display("Estados FSM Cajero Automatico \n");
21     $display("CLK Reset | on push1 push2 password checks|ON Logeado Displays_Login Retiro Consulta Sin_Dinero Saldo");
22     $monitor("%b %b | %b %b %b %b %b | %b %b %b %b %b %b %b %b %b %b",
23         clock, reset, on, push1, push2, password, checks, Encendido, Login, Display_Login, pulso, Display_Saldo, Vacio, Saldo);
24     #1 reset = 1; // se realiza el reseteo inicial de la máquina de moore
25     #2 on=0; push1=0; push2=0; password[0]=0;password[1]=0;password[2]=0;password[3]=0; checks=000; reset = 0;
26     #2 on=0; push1=1; push2=0; password[0]=0;password[1]=0;password[2]=0;password[3]=0; checks=000;
27     #2 on=0; push1=0; push2=1; password[0]=0;password[1]=0;password[2]=0;password[3]=0; checks=000;
28     #2 on=1; push1=0; push2=0; password[0]=0;password[1]=0;password[2]=0;password[3]=0; checks=000;

```

Figura 35. Código en verilog 6

```

Project: maquina_cajero.v X maquina_cajero_tb.v X
programacion
  .consigna.dble
  .api.ini
  maquina_cajero_tb.v
  maquina_cajero_vcd
  maquina_cajero.v
  maquina_cajero_tb.vcd

49 #2 on=1; push1=0; push2=1; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //prueba consulta
50
51 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
52 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
53 #2 on=1; push1=1; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
54 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
55 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
56 #2 on=1; push1=1; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
57 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
58 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
59 #2 on=1; push1=1; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
60 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
61 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
62 #2 on=1; push1=1; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
63 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
64 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
65 #2 on=1; push1=1; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //prueba de retro para vaciar cajero
66 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
67 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
68
69 #2 on=1; push1=0; push2=1; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //prueba de consulta última
70 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
71 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
72
73 #2 on=1; push1=1; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111;
74 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //delay
75 #2 on=1; push1=0; push2=0; password[0]=1;password[1]=0;password[2]=0;password[3]=1; checks=111; //prueba de vaciar cajero
76 #1 $finish; //finalizar prueba de cajero
77
78 end
79
80 initial begin
81   $dumpfile("maquina_cajero_tb.vcd"); // se ejecuta GTKwave
82   $dumpvars(0, testbench);
83 end
84
85 endmodule
86

```

Figura 37. Código en verilog 8

CLK	Reset	on	push1	push2	password	checks	ON	Logeado	Displays_Login	Retiro	Consulta	Sin_Dinero	Saldo
VCD	info:	dumpfile	maquina_cajero_tb.vcd	opened	for	output.							
1	x	x	x	x	XXXX	XXX	x	x	XXX	x	x	0	111
0	1	x	x	x	XXXX	XXX	0	0	000	0	0	0	111
1	1	x	x	x	XXXX	XXX	0	0	000	0	0	0	111
0	0	0	0	0	0000	000	0	0	000	0	0	0	111
1	0	0	0	0	0000	000	0	0	000	0	0	0	111
0	0	0	1	0	0000	000	0	0	000	0	0	0	111
1	0	0	1	0	0000	000	0	0	000	0	0	0	111
0	0	0	0	1	0000	000	0	0	000	0	0	0	111
1	0	0	0	1	0000	000	0	0	000	0	0	0	111
0	0	1	0	0	0000	000	0	0	000	0	0	0	111
1	0	1	0	0	0000	000	1	0	000	0	0	0	111
0	0	1	1	0	0000	000	1	0	000	0	0	0	111
1	0	1	1	0	0000	000	1	0	000	0	0	0	111
0	0	1	0	1	0000	000	1	0	000	0	0	0	111

Figura 38. Pruebas en consola con diferentes combinaciones 1

0	0	1	0	0	0010	011	1	0	000	0	0	0	111
1	0	1	0	0	0010	011	1	0	000	0	0	0	111
0	0	1	0	0	0101	111	1	0	000	0	0	0	111
1	0	1	0	0	0101	111	1	0	000	0	0	0	111
0	0	1	0	0	0010	001	1	0	000	0	0	0	111
1	0	1	0	0	0010	001	1	0	001	0	0	0	111
0	0	1	0	0	0110	011	1	0	001	0	0	0	111
1	0	1	0	0	0110	011	1	0	011	0	0	0	111
0	0	1	0	0	0101	111	1	0	011	0	0	0	111
1	0	1	0	0	0101	111	1	0	011	0	0	0	111
0	0	1	0	0	0010	001	1	0	011	0	0	0	111
1	0	1	0	0	0010	001	1	0	011	0	0	0	111
0	0	1	0	0	0110	011	1	0	011	0	0	0	111
1	0	1	0	0	0110	011	1	0	011	0	0	0	111
0	0	1	0	0	1001	111	1	0	011	0	0	0	111
1	0	1	0	0	1001	111	1	1	111	0	0	0	111
0	0	1	0	0	1001	111	1	1	111	0	0	0	111
1	0	1	0	0	1001	111	1	1	111	0	0	0	111
0	0	1	1	0	1001	111	1	1	111	0	0	0	111
1	0	1	1	0	1001	111	1	1	111	1	0	0	111

Figura 39. Pruebas en consola con diferentes combinaciones 2

1	0	1	0	1001	111	1	1	111	0	0	0	110
0	0	1	1	1001	111	1	1	111	0	0	0	110
1	0	1	1	1001	111	1	1	111	1	0	0	110
0	0	1	0	1001	111	1	1	111	1	0	0	110
1	0	1	0	1001	111	1	1	111	0	0	0	101
0	0	1	0	1001	111	1	1	111	0	0	0	101
1	0	1	0	1001	111	1	1	111	0	0	0	101
0	0	1	1	1001	111	1	1	111	0	0	0	101
1	0	1	1	1001	111	1	1	111	1	0	0	101
0	0	1	0	1001	111	1	1	111	0	0	0	100
1	0	1	0	1001	111	1	1	111	0	0	0	100
0	0	1	1	1001	111	1	1	111	0	0	0	100
1	0	1	1	1001	111	1	1	111	1	0	0	100
0	0	1	0	1001	111	1	1	111	0	0	0	100
1	0	1	0	1001	111	1	1	111	0	0	0	011
0	0	1	0	1001	111	1	1	111	0	0	0	011
1	0	1	0	1001	111	1	1	111	0	0	0	011
0	0	1	1	1001	111	1	1	111	0	0	0	011
1	0	1	1	1001	111	1	1	111	1	0	0	011
0	0	1	0	1001	111	1	1	111	1	0	0	011
1	0	1	0	1001	111	1	1	111	0	0	0	010
0	0	1	0	1001	111	1	1	111	0	0	0	010
1	0	1	0	1001	111	1	1	111	0	0	0	010
0	0	1	1	1001	111	1	1	111	0	0	0	010
1	0	1	1	1001	111	1	1	111	1	0	0	010
0	0	1	0	1001	111	1	1	111	1	0	0	010

Figura 40. Pruebas en consola con diferentes combinaciones 4

1	0	1	0	1001	111	1	1	111	0	0	0	001
0	0	1	0	1001	111	1	1	111	0	0	0	001
1	0	1	0	1001	111	1	1	111	0	0	0	001
0	0	1	0	1001	111	1	1	111	0	0	0	001
1	0	1	0	1001	111	1	1	111	0	1	0	001
0	0	1	0	1001	111	1	1	111	0	1	0	001
1	0	1	0	1001	111	1	1	111	0	0	0	001
0	0	1	0	1001	111	1	1	111	0	0	0	001
1	0	1	0	1001	111	1	1	111	0	0	0	001
0	0	1	1	1001	111	1	1	111	0	0	0	001
1	0	1	1	1001	111	1	1	111	1	0	0	001
0	0	1	0	1001	111	1	1	111	1	0	0	001
1	0	1	0	1001	111	1	1	111	0	0	1	000
0	0	1	0	1001	111	1	1	111	0	0	1	000
1	0	1	0	1001	111	1	1	111	0	0	1	000

Figura 41. Pruebas en consola con diferentes combinaciones 5

IX. Implementación en CircuitVerse

Se realizó la implementación de cada uno de las FSM y la máquina en conjunto en circuitverse:

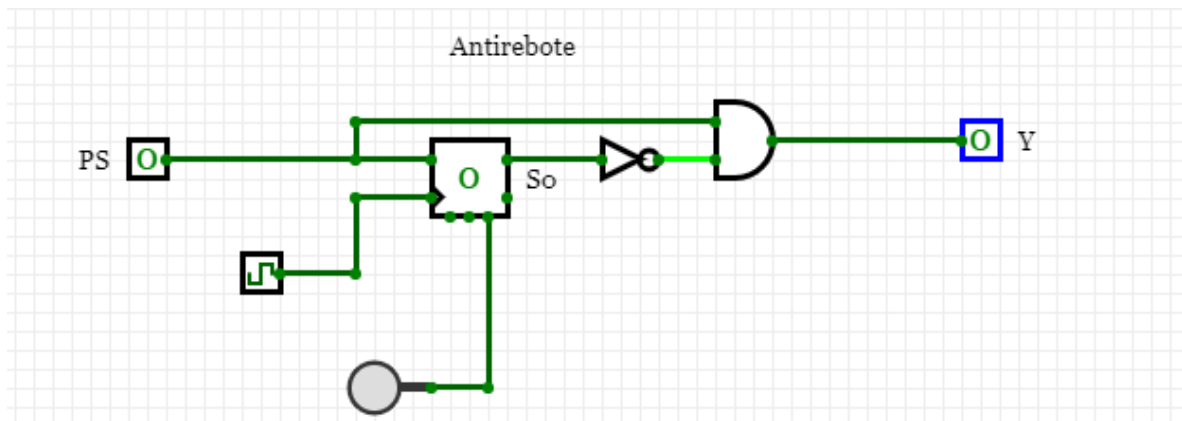


Figura 42. Implementación en circuitverse FSM antirebote

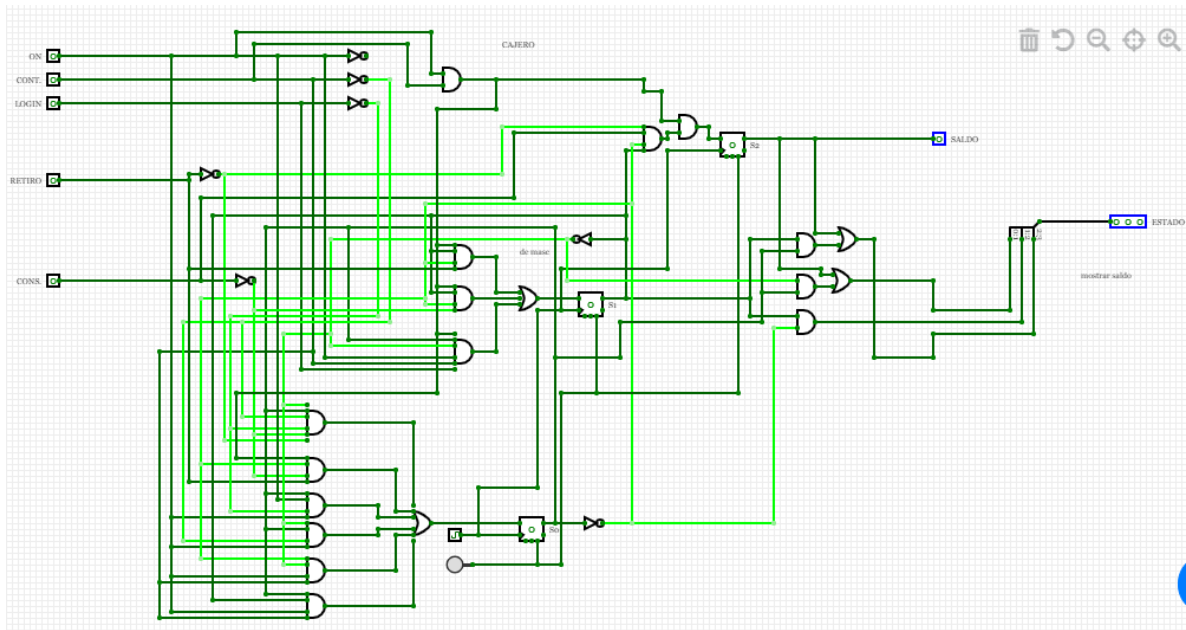


Figura 45. Implementación en circuitverse FSM Cajero

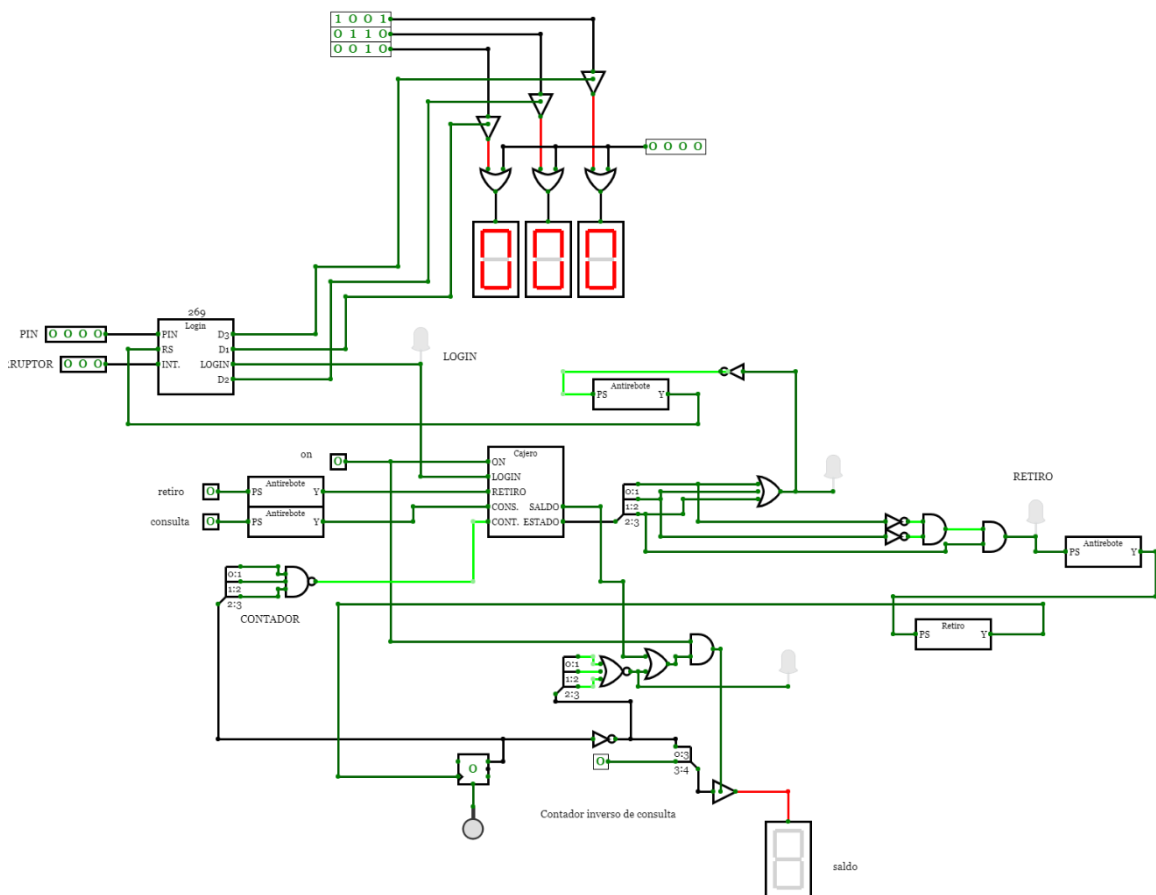


Figura 46. Implementación en circuitverse Máquina completa Cajero automático

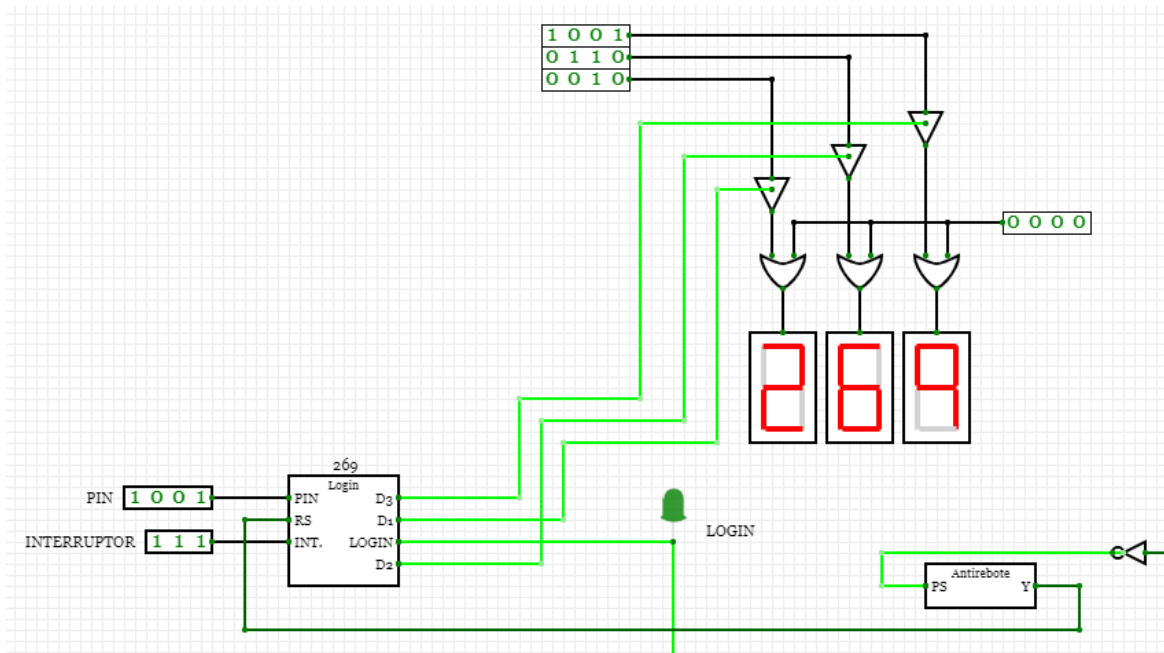


Figura 47. Funcionamiento de verificación

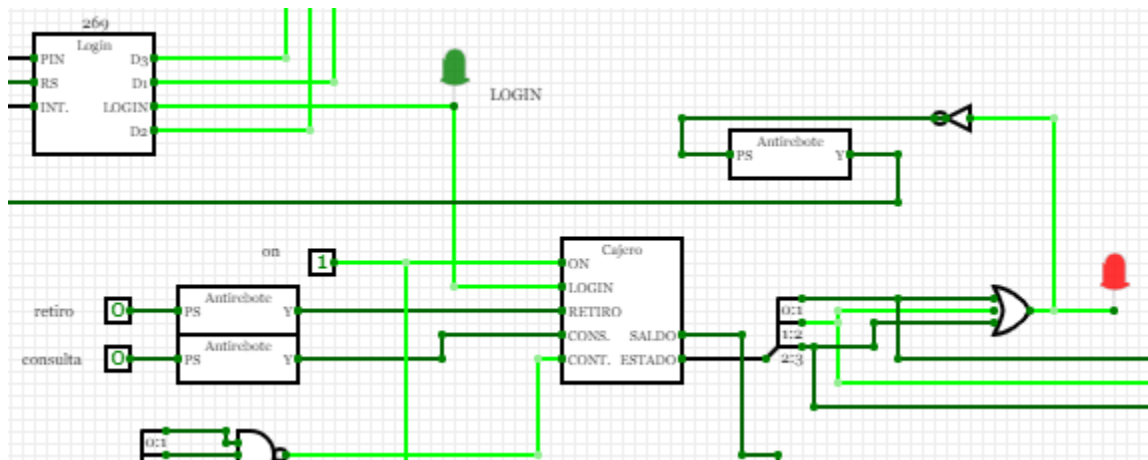


Figura 48. Funcionamiento de cajero

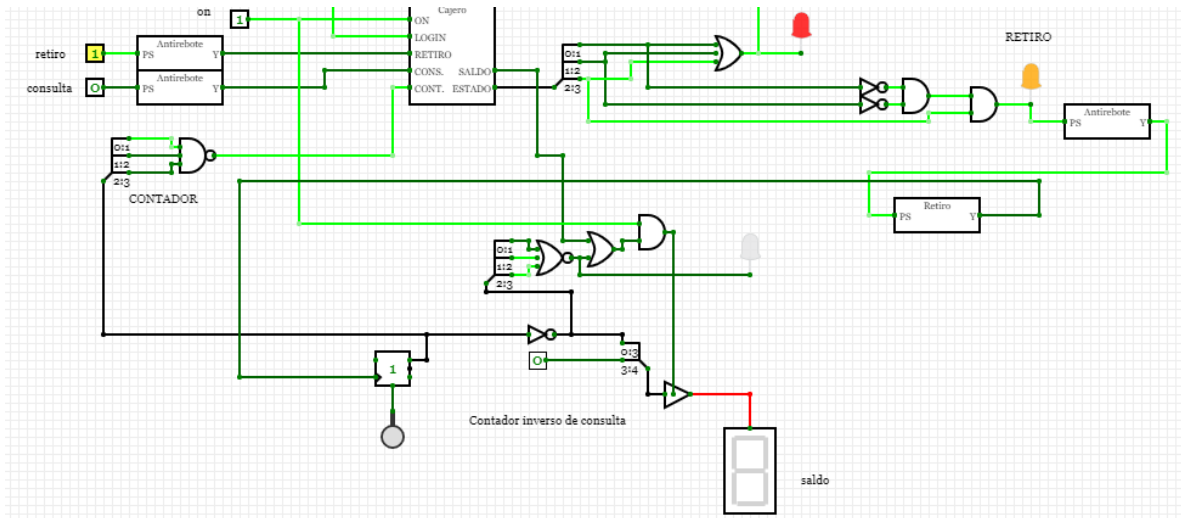


Figura 49. Funcionamiento de retiro

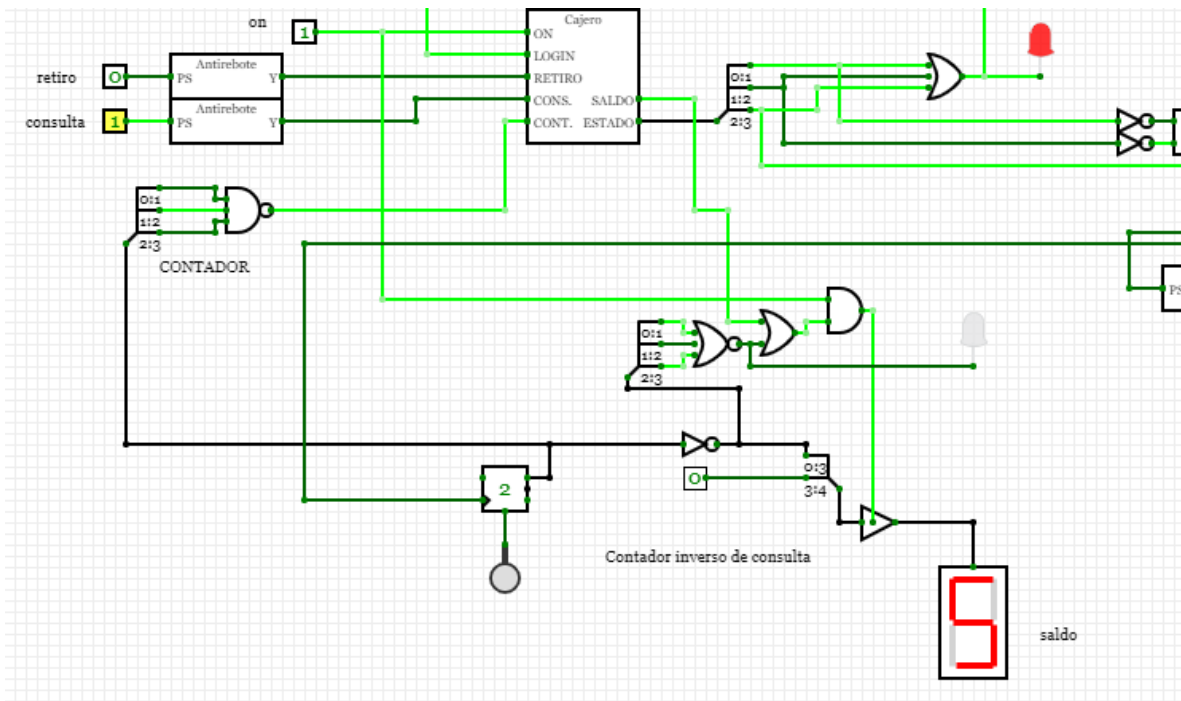


Figura 50. Funcionamiento de consulta

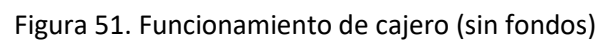


Figura 51. Funcionamiento de cajero (sin fondos)