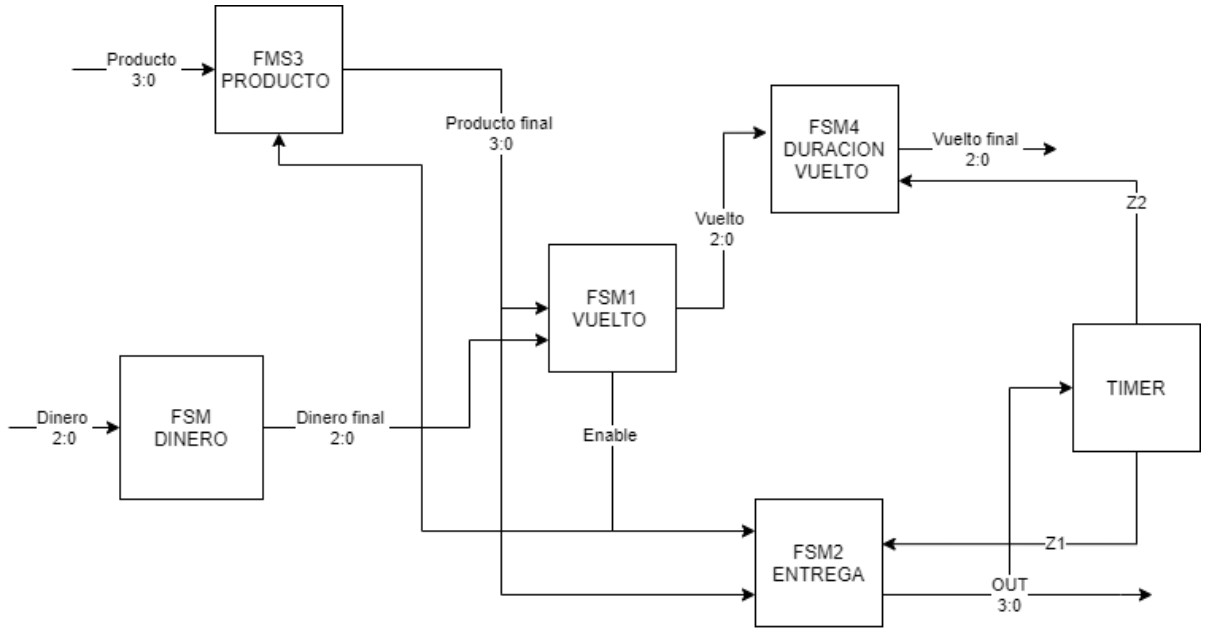
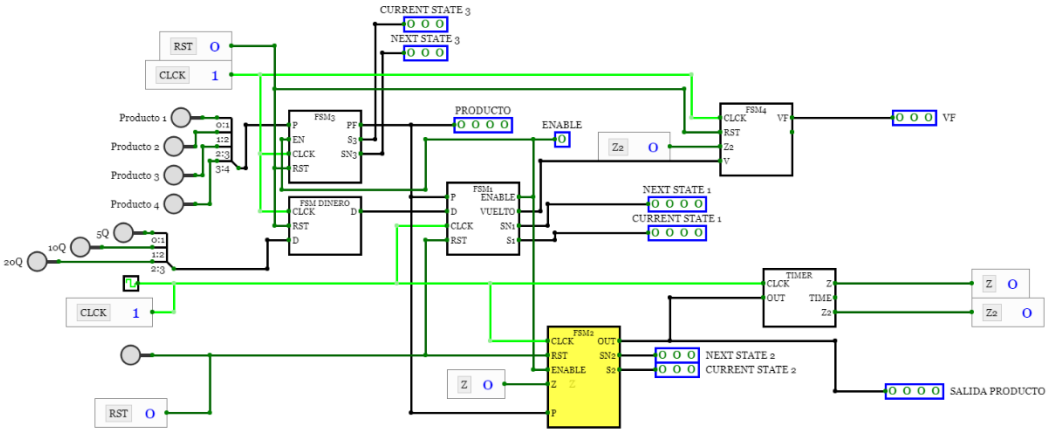
Carlos Gil 19443

Proyecto 1 Maquina expendedora

Explicacion general y cajas negras:

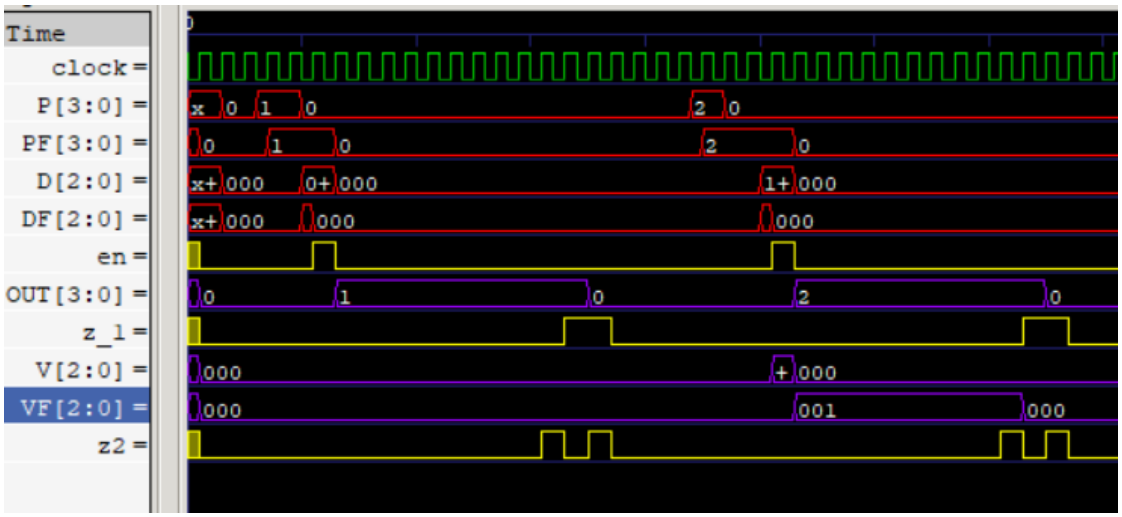




El proyecto consiste en realizar una maquina expendedora con la opción de escoger entre 4 productos y pagarlos con billetes de Q5, Q10 y Q20. También, tendrá la opción de dar vuelo en caso de que el monto ingresado, exceda el valor del producto. Dicho vuelo, únicamente puede ser Q5, Q10 o Q20, en caso de que el vuelto sea una cantidad distinta a estas, la maquina escogerá entre las 3 opciones mencionadas anteriormente, la que mas se acerque a la cantidad real y conservara el resto. Al momento de entregar el producto, esta salida tendrá una duración mayor a un ciclo de reloj, de modo que se pueda simular la acción que realiza una maquina real cuando entrega un producto.

Esta funcionara de la siguiente manera, inicialmente se escogerá cual de los productos se desea, luego se esperará que ingrese la cantidad de dinero correspondiente, si este dinero no es suficiente, la maquina no autorizara que se le sea brindado el producto hasta que lo sea y si este se excede, la maquina le devolverá la cantidad mas cercana a su vuelto con un billete de Q5 o Q10 o Q20. Una vez se detecte que el pago fue correcto, la maquina procederá a brindarle su producto y el proceso volverá a iniciar.

Diagramas de timing e implementación en verilog:



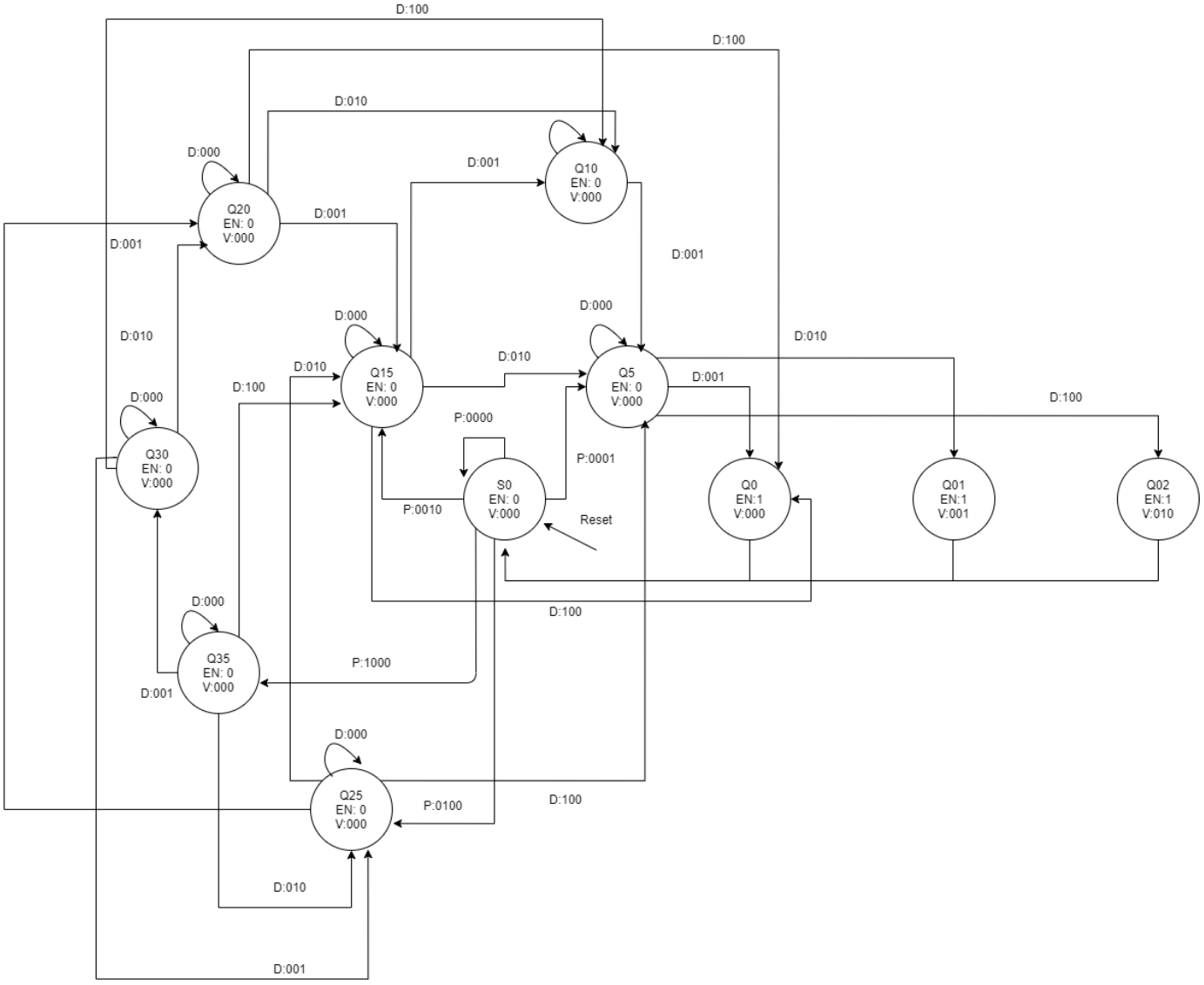
En el diagrama se puede observar que cuando P (Producto) recibe un pulso en la entrada se selecciona el producto deseado, luego entra a PF que se mantiene activa hasta que EN vuelve a ser 0. La entrada D entra a un antirebote cuya salida es DF, de modo que la máquina de cobro únicamente realice un cobro con el billete ingresado. EN se colca en uno cuando la maquina de cobro registra que ya se termino de pagar, con ello el producto seleccionado pasa a la salida (OUT) y esta salida esta activa hasta que z_1 se activa, lo cual sucede cuando el timer llega a 10. En el caso de la V, esta salida es activada por la maquina de cobro cuando al cliente le corresponde un vuelto, en este caso no se activo ya que como se ve en D, se pago con 5Q lo que corresponde al precio del producto 1. Sin embargo se puede observar que z2 tiene un pulso, este sucede cuando el TIMER llega a 9, este indica que la duración del vuelto, de modo que la salida se resetea cuando z2 se enciende.

Explicación individual de cada una de las máquinas de estados finitos

FSM1 - vuelto:

Esta máquina de asignarle al cliente una cantidad a pagar acorde con el producto que selecciono e ir restando esa cantidad conforme el cliente va ingresando dinero. Así mismo, cunado el cliente termina de ingresar el dinero autoriza que el producto sea despachado al cliente y le da vuelto en el caso de que lo corresponda.

Diagrama de estados:



Codificación de las entradas, salidas y estados:

Estados
S0 = 0000
Q5 = S1 = 0001
Q10 = S2 = 0010
Q15 = S3 = 0011
Q20 = S4 = 0100
Q25 = S5 = 0101
Q30 = S6 = 0110
Q35 = S7 = 0111
Q0 = S8 = 1000
Q01 = S9 =1001
Q02 = S10 = 1010

Dinero = D
Q5 = 001
Q10 = 010
Q20 = 100
Vuelto = V
Q5 = 001
Q10 = 010
Q20 = 100

Productos = P
Producto 1 = 0001 = Q5
Producto 2 = 0010 = Q15
Producto 3 = 0100 = Q25
Producto 4 = 1000 = Q35

Tablas codficadas:

Estados:

S3	S2	S1	S0	P3	P2	P1	P0	D2	D1	D0		SN3	SN2	SN1	SN0
0	0	0	0	0	0	0	0	X	X	X		0	0	0	0
0	0	0	0	0	0	0	1	X	X	X		0	0	0	1
0	0	0	0	0	0	1	0	X	X	X		0	0	1	1
0	0	0	0	0	1	0	0	X	X	X		0	1	0	1
0	0	0	0	1	0	0	0	X	X	X		0	1	1	1
0	0	0	1	X	X	X	X	0	0	0		0	0	0	1
0	0	0	1	X	X	X	X	0	0	1		1	0	0	0
0	0	0	1	X	X	X	X	0	1	0		1	0	0	1
0	0	0	1	X	X	X	X	1	0	0		1	0	1	0
0	0	1	0	X	X	X	X	0	0	0		0	0	1	0
0	0	1	0	X	X	X	X	0	0	1		0	0	0	1
0	0	1	0	X	X	X	X	0	1	0		1	0	0	0
0	0	1	0	X	X	X	X	1	0	0		1	0	1	0
0	0	1	1	X	X	X	X	0	0	0		0	0	1	1
0	0	1	1	X	X	X	X	0	0	1		0	0	1	0
0	0	1	1	X	X	X	X	0	1	0		0	0	0	1
0	0	1	1	X	X	X	X	1	0	0		1	0	0	1
0	1	0	0	X	X	X	X	0	0	0		0	1	0	0
0	1	0	0	X	X	X	X	0	0	1		0	0	1	1
0	1	0	0	X	X	X	X	0	1	0		0	0	1	0
0	1	0	0	X	X	X	X	1	0	0		1	0	0	0
0	1	0	1	X	X	X	X	0	0	0		0	1	0	1
0	1	0	1	X	X	X	X	0	0	1		0	1	0	0
0	1	0	1	X	X	X	X	0	1	0		0	0	1	1
0	1	0	1	X	X	X	X	1	0	0		0	0	0	1
0	1	1	0	X	X	X	X	0	0	0		0	1	1	0
0	1	1	0	X	X	X	X	0	0	1		0	1	0	1
0	1	1	0	X	X	X	X	0	1	0		0	1	0	0
0	1	1	0	X	X	X	X	1	0	0		0	0	1	0
0	1	1	1	X	X	X	X	0	0	0		0	1	1	1
0	1	1	1	X	X	X	X	0	0	1		0	1	1	0
0	1	1	1	X	X	X	X	0	1	0		0	1	0	1
0	1	1	1	X	X	X	X	1	0	0		0	0	1	1
1	0	0	0	X	X	X	X	X	X	X		0	0	0	0
1	0	0	1	X	X	X	X	X	X	X		0	0	0	0
1	0	1	0	X	X	X	X	X	X	X		0	0	0	0
1	0	1	1	X	X	X	X	X	X	X		X	X	X	X
1	1	0	0	X	X	X	X	X	X	X		X	X	X	X
1	1	0	1	X	X	X	X	X	X	X		X	X	X	X
1	1	1	0	X	X	X	X	X	X	X		X	X	X	X
1	1	1	1	X	X	X	X	X	X	X		X	X	X	X

Salidas:

S3	S2	S1	S0		EN	V2	V1	V0
0	0	0	0		0	0	0	0
0	0	0	0		0	0	0	0
0	0	0	0		0	0	0	0
0	0	0	0		0	0	0	0
0	0	0	0		0	0	0	0
0	0	0	1		0	0	0	0
0	0	0	1		0	0	0	0
0	0	0	1		0	0	0	0
0	0	0	1		0	0	0	0
0	0	1	0		0	0	0	0
0	0	1	0		0	0	0	0
0	0	1	0		0	0	0	0
0	0	1	0		0	0	0	0
0	0	1	1		0	0	0	0
0	0	1	1		0	0	0	0
0	0	1	1		0	0	0	0
0	0	1	1		0	0	0	0
0	1	0	0		0	0	0	0
0	1	0	0		0	0	0	0
0	1	0	0		0	0	0	0
0	1	0	0		0	0	0	0
0	1	0	1		0	0	0	0
0	1	0	1		0	0	0	0
0	1	0	1		0	0	0	0
0	1	0	1		0	0	0	0
0	1	1	0		0	0	0	0
0	1	1	0		0	0	0	0
0	1	1	0		0	0	0	0
0	1	1	0		0	0	0	0
0	1	1	1		0	0	0	0
0	1	1	1		0	0	0	0
0	1	1	1		0	0	0	0
0	1	1	1		0	0	0	0
1	0	0	0		1	0	0	0
1	0	0	1		1	0	0	1
1	0	1	0		1	0	1	0
1	0	1	1		X	X	X	X
1	1	0	0		X	X	X	X
1	1	0	1		X	X	X	X
1	1	1	0		X	X	X	X
1	1	1	1		X	X	X	X

Ecuaciones booleanas:

Estados:

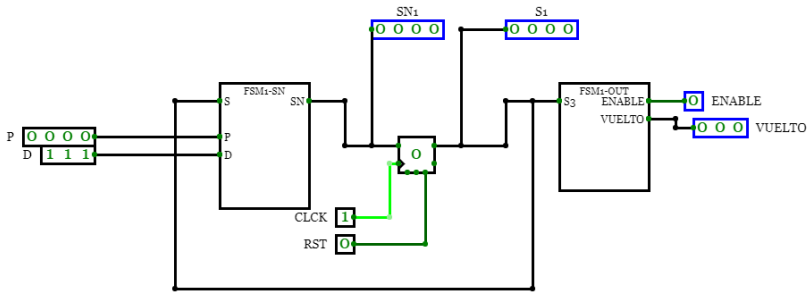
```
Minimized:
SN3 = S3' S2' S1 D2 D1' D0' + S3' S2' S0 D2 D1' D0' + S2 S1' S0' D2 D1' D0' + S3' S2' S1' S0 D2' D1' D0 + S3' S2' S1' S0 D2' D1
D0' + S3' S2' S1 S0' D2' D1 D0';
SN2 = S2 S1 D2' D1'  + S2 S0 D2' D1'  + S2 S1 D2' D0' + S2 D2' D1' D0' + S3' S2' S1' S0' P3' P2 P1' P0'  + S3' S2' S1' S0' P3 P2'
P1' P0' ;
SN1 = S1 S0 D2' D1'  + S2 S1 D1' D0' + S2 S1' D2' D1 D0' + S3' S1 S0' D1' D0' + S2 S1' S0' D2' D1' D0 + S3' S2' S1' S0 D2 D1' D0'
+ S3' S2' S1' S0' P3' P2' P1 P0'  + S3' S2' S1' S0' P3 P2' P1' P0' ;
SN0 = S3' S0 D2' D0' + S2 S0 D1' D0' + S1 S0 D1' D0' + S2 S0' D2' D1' D0 + S3' S1 S0' D2' D1' D0 + S3' S2' S1' S0' P3' P2' P1' P0
+ S3' S2' S1' S0' P3' P2' P1 P0'  + S3' S2' S1' S0' P3' P2 P1' P0'  + S3' S2' S1' S0' P3 P2' P1' P0' ;
```

Salidas:

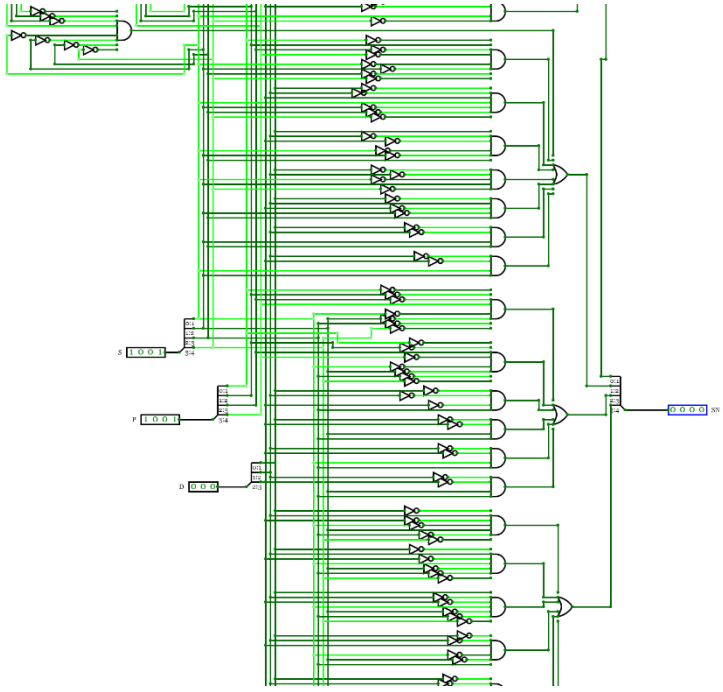
```
Entered by truthtable:
EN = S3 S2' S1' S0' + S3 S2' S1' S0 + S3 S2' S1 S0';
V2 = 0;
V1 = S3 S2' S1 S0';
V0 = S3 S2' S1' S0;
```

Implementación en circuitverse:

FSM:



Estados:



Salidas:

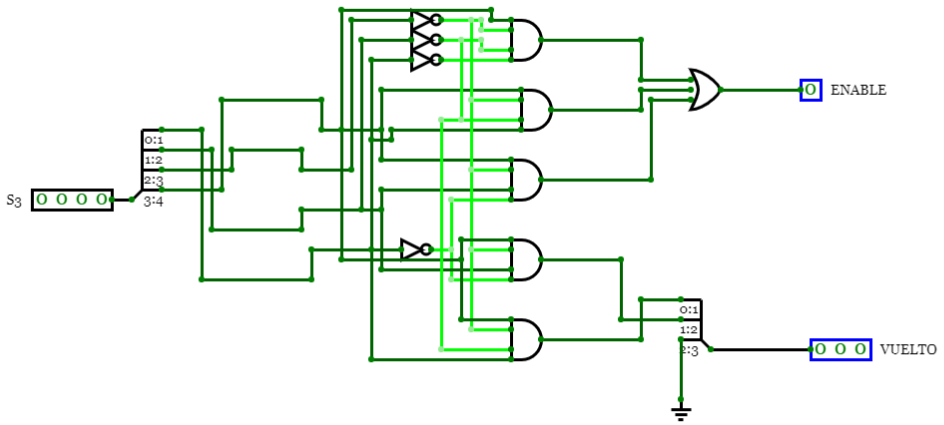
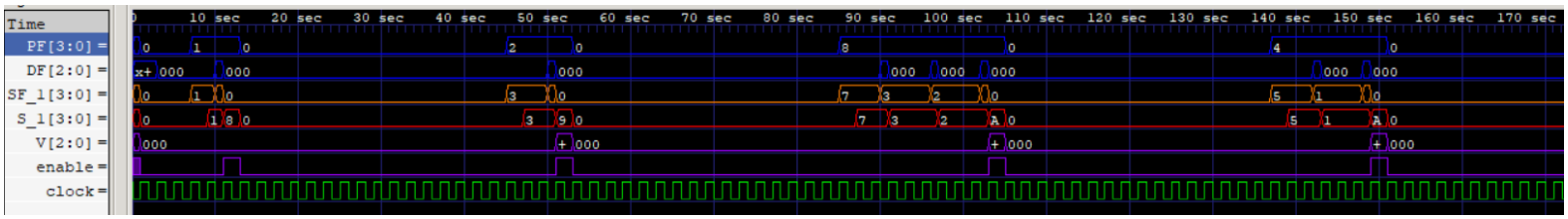


Diagrama de timing e implementacion en verilog:

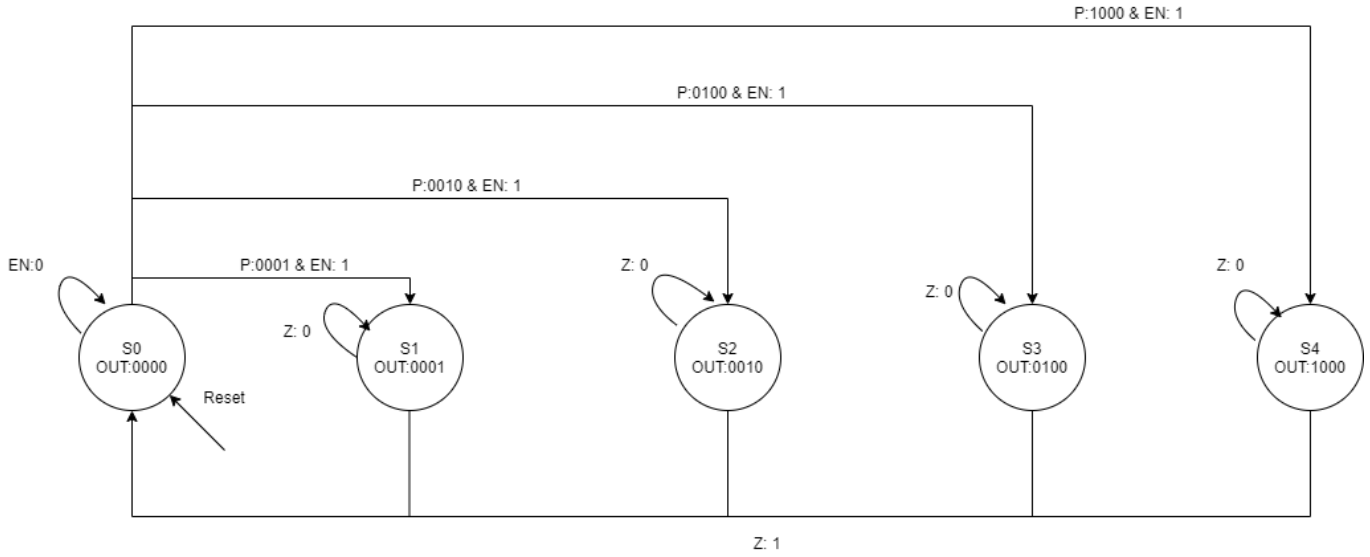


En el diagrama se puede observar que al momento que PF tiene un valor, se procede a salir del estado 0 al estado de deuda que corresponda con el producto seleccionado, luego en ese estado unicamente se sale al ingresar un valor de dinero, en el caso de la primera prueba se observa que DF es 001 indicando 5Q, como este es el precio justo del producto escogido, se cambia de estado a un estado cuya salida es coloca el ENABLE en 1 y asignar un vuelto, el cual puede ser 0 o 5 o 10 Q.

FMS2 - maquina para repartir productos:

Esta maquina se encarga de recibir la autorizacion de pago cuando el cliente finaliza de pagar y entrega el producto que fue seleccionado por el cliente. Asi mismo, para simular los motores que hacen girar la espiral con la que se despachan los productos en una maquina real, la salida entra a un timer para que esta permanezca activa durante 10 ciclos de reloj antes de reiniciarse.

Diagramas de estado:



Codificación de entradas, salidas y estados:

EN
Z

Productos = P	Estados:	OUT = Producto
Producto 1 = 0001 = Q5	S0 = 000	Producto 1 = 0001
Producto 2 = 0010 = Q15	S1 = 001	Producto 2 = 0010
Producto 3 = 0100 = Q25	S2 = 010	Producto 3 = 0100
Producto 4 = 1000 = Q35	S3 = 011	Producto 4 = 1000
	S4 = 100	

Tablas codificadas:

Estados:

S1	S1	S0	P3	P2	P1	P0	Z	EN		SN2	SN1	SN0
0	0	0	X	X	X	X	X	0		0	0	0
0	0	0	0	0	0	1	X	1		0	0	1
0	0	0	0	0	1	0	X	1		0	1	0
0	0	0	0	1	0	0	X	1		0	1	1
0	0	0	1	0	0	0	X	1		1	0	0
0	0	1	X	X	X	X	0	X		0	0	1
0	0	1	X	X	X	X	1	X		0	0	0
0	1	0	X	X	X	X	0	X		0	1	0
0	1	0	X	X	X	X	1	X		0	0	0
0	1	1	X	X	X	X	0	X		0	1	1
0	1	1	X	X	X	X	1	X		0	0	0
1	0	0	X	X	X	X	0	X		1	0	0
1	0	0	X	X	X	X	1	X		0	0	0

Salidas:

S2	S1	S0		O3	O2	O1	O0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	1		0	0	0	1
0	0	1		0	0	0	1
0	1	0		0	0	1	0
0	1	0		0	0	1	0
0	1	1		0	1	0	0
0	1	1		0	1	0	0
1	0	0		1	0	0	0
1	0	0		1	0	0	0

Ecuaciones booleanas:

Estados:

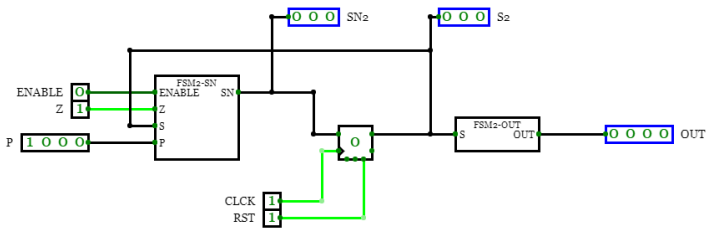
```
Minimized:
SN2 = S2 S1' S0' Z'  + S2' S1' S0' P3 P2' P1' P0' EN;
SN1 = S2' S1 Z'  + S2' S1' S0' P3' P2' P1 P0' EN + S2' S1' S0' P3' P2 P1' P0' EN;
SN0 = S2' S0 Z'  + S2' S1' S0' P3' P2' P1' P0 EN + S2' S1' S0' P3' P2 P1' P0' EN;
```

Salidas:

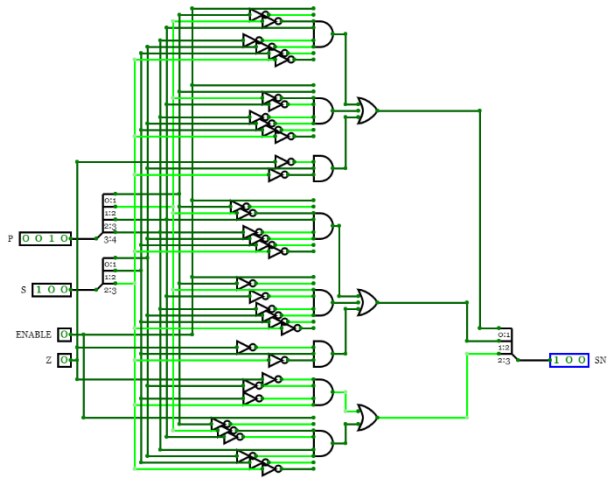
```
Minimized:
O3 = S2 S1' S0';
O2 = S2' S1 S0;
O1 = S2' S1 S0';
O0 = S2' S1' S0;
```

Implementacion en circuitverse

FSM:



Estados:



Salidas:

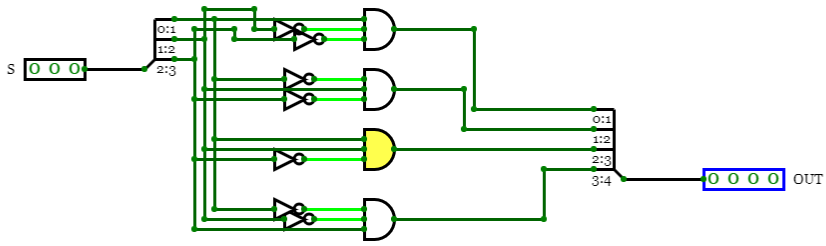
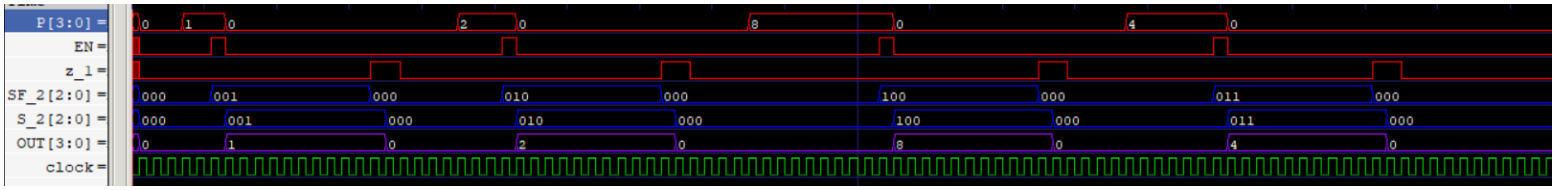


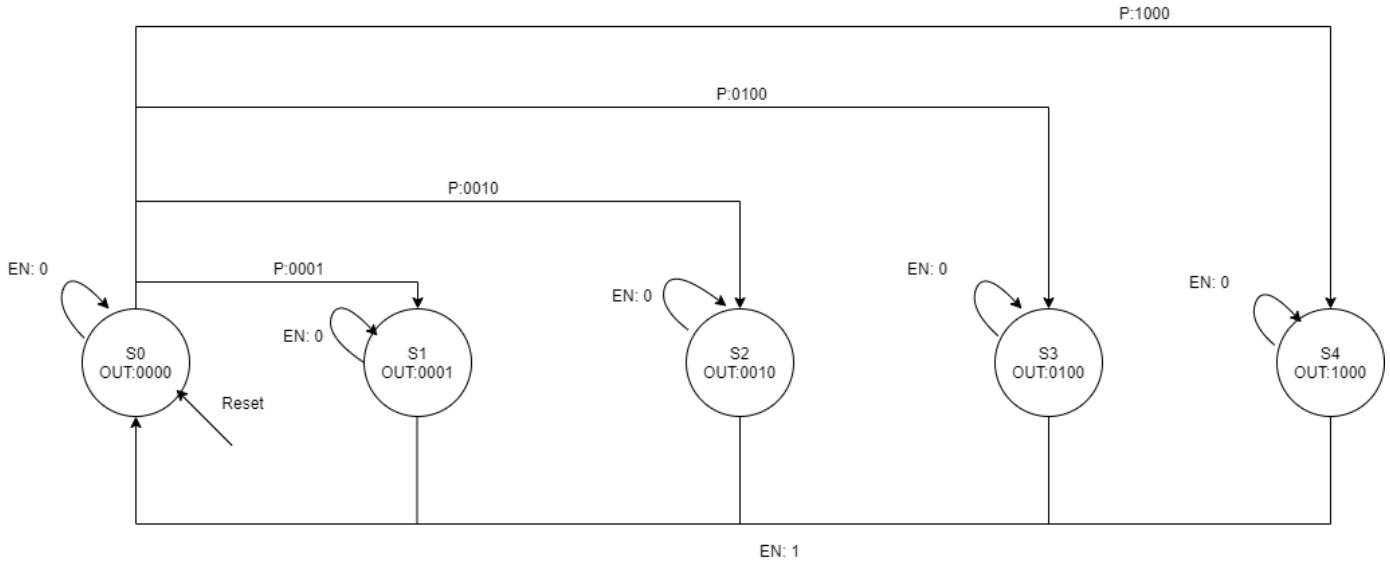
Diagrama de timing e implementación en verilog



En el diagrama se puede observar que mientras EN es 0, no se sale del estado inicial, al momento de que este sea 1, inidcando que ya se termino de pagar, se pasa al siguiente estado cuya salida corresponde a el producto seleccionado por el usuario. La salida se mantiene activa hasta que z_1 es uno, indicando que el timer llego a 10 de modo que la salida ya se puede resetear a la espera de otro cliente.

FSM3 – Selector de productos

Esta maquina se encarga de mantener activo el valor ingresado con los push bottons hasta que la maquina de cobro detecte que el cliente ya termino de pagar el producto solicitado. Esto con el fin de que el input de producto ingresado por el cliente continue, para poder entrar a la maquina de entrega y repartir el producto correspondiente.



Codificación de las entradas, salidas y estados:

EN

Estados:
S0 = 000
S1 = 001
S2 = 010
S3 = 011
S4 = 100

OUT = Producto
Producto 1 = 0001
Producto 2 = 0010
Producto 3 = 0100
Producto 4 = 1000

Tablas codificadas:

Estados:

S1	S1	S0	P3	P2	P1	P0	EN		SN2	SN1	SN0
0	0	0	X	X	X	X	0		0	0	0
0	0	0	0	0	0	1	X		0	0	1
0	0	0	0	0	1	0	X		0	1	0
0	0	0	0	1	0	0	X		0	1	1
0	0	0	1	0	0	0	X		1	0	0
0	0	1	X	X	X	X	0		0	0	1
0	0	1	X	X	X	X	1		0	0	0
0	1	0	X	X	X	X	0		0	1	0
0	1	0	X	X	X	X	1		0	0	0
0	1	1	X	X	X	X	0		0	1	1
0	1	1	X	X	X	X	1		0	0	0
1	0	0	X	X	X	X	0		1	0	0
1	0	0	X	X	X	X	1		0	0	0

Salidas:

S2	S1	S0		O3	O2	O1	O0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	0		0	0	0	0
0	0	1		0	0	0	1
0	0	1		0	0	0	1
0	1	0		0	0	1	0
0	1	0		0	0	1	0
0	1	1		0	1	0	0
0	1	1		0	1	0	0
1	0	0		1	0	0	0
1	0	0		1	0	0	0

Ecuaciones booleanas:

Estados:

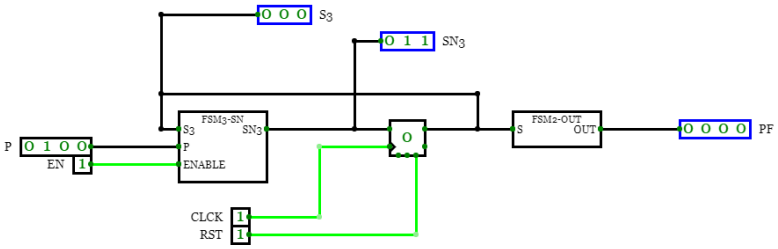
```
Minimized:
SN2 = S2 EN' + S2' S1' S0' P3 P2' P1' P0' ;
SN1 = S1 EN' + S2' S1' S0' P3' P2 P1' P0'  + S2' S1' S0' P3' P2' P1 P0' ;
SN0 = S0 EN' + S2' S1' S0' P3' P2 P1' P0'  + S2' S1' S0' P3' P2' P1' P0 ;
```

Salidas:

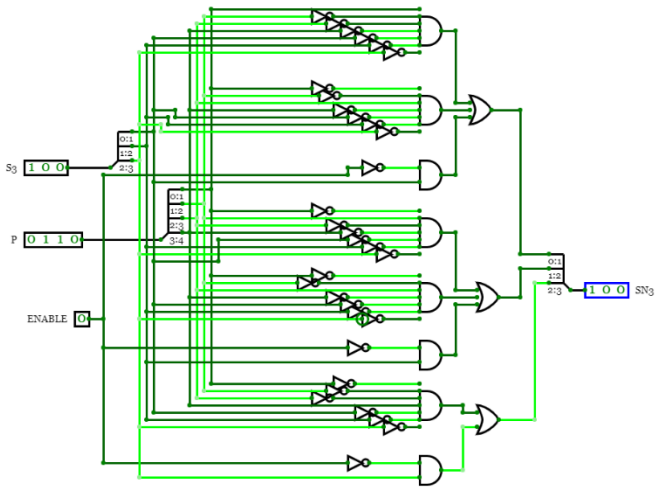
```
Minimized:
O3 = S2 S1' S0';
O2 = S2' S1 S0;
O1 = S2' S1 S0';
O0 = S2' S1' S0;
```

Implementación en circuitverse:

FSM:



Estados:



Salidas:

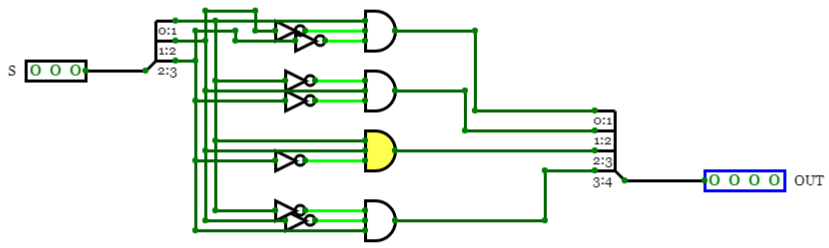
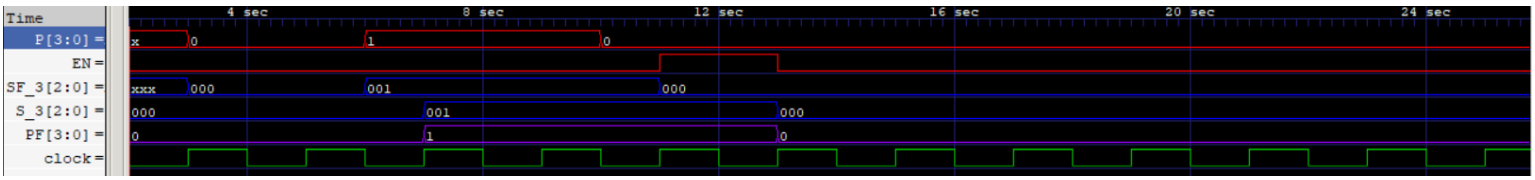


Diagrama de timing e implementacion en verilog:

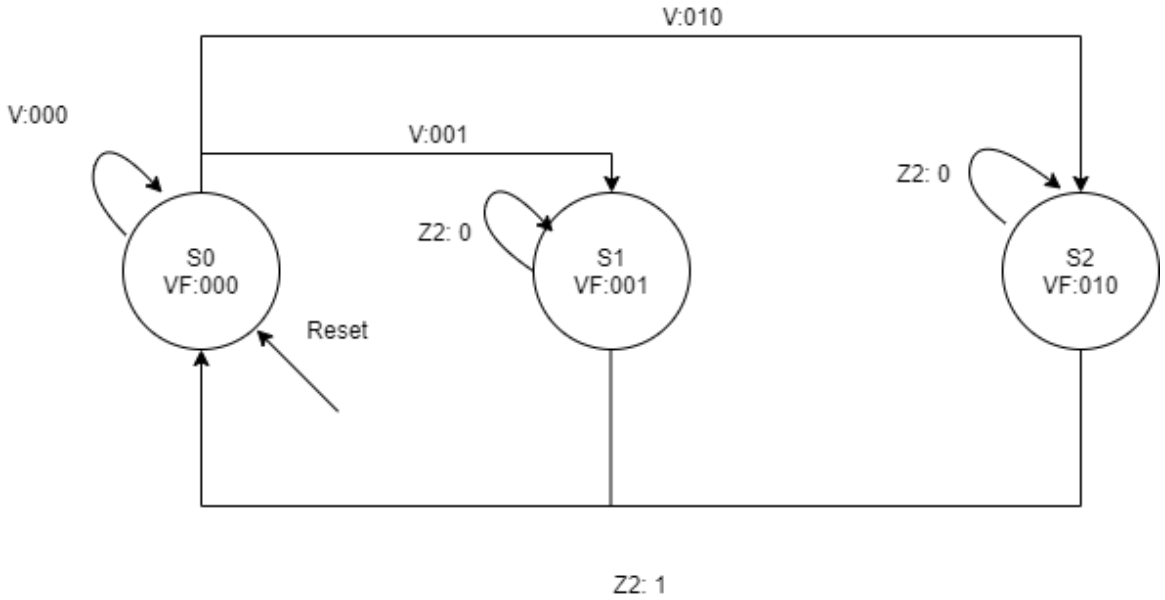


En el diagrama se puede observar que al momento de ingresar un producto P, se cambia de estado cuya salida es el mismo producto. Esta salida se mantiene hasta que el EN se coloca en 1 y cuando este vuelve a ser 0 la maquina regresa el estado inicial con salida 0 esperando que otro producto sea ingresado.

FSM4 – Duracion del vuelto

De forma muy similar que la maquina de entrega el producto, esta maquina se encarga de hacer que la salida del vuelto tenga una duracion mayor a un ciclo de reloj, de modo que pueda simular un motor que este encargado de sacar el billete de la maquina, como una maquina real.

Diagrama de estados:



Codificacion de entradas, salidas y estados:

Z2
Estados:
S0 = 00
S1 = 01
S2 = 10

Vuelto = V = VF
Q5 = 001
Q10 = 010
Q20 = 100

Tablas codificadas:

Estados:

S1	S0	V2	V1	V0	Z		SN1	SN0
0	0	0	0	0	X		0	0
0	0	0	0	1	X		0	1
0	0	0	1	0	X		1	0
0	1	X	X	X	0		0	1
0	1	X	X	X	1		0	0
1	0	X	X	X	0		1	0
1	0	X	X	X	1		0	0

Salidas:

S1	S0		VF2	VF1	VF0
0	0		0	0	0
0	0		0	0	0
0	0		0	0	0
0	1		0	0	1
0	1		0	0	1
1	0		0	1	0
1	0		0	1	0

Ecuaciones booleanas:

Estados:

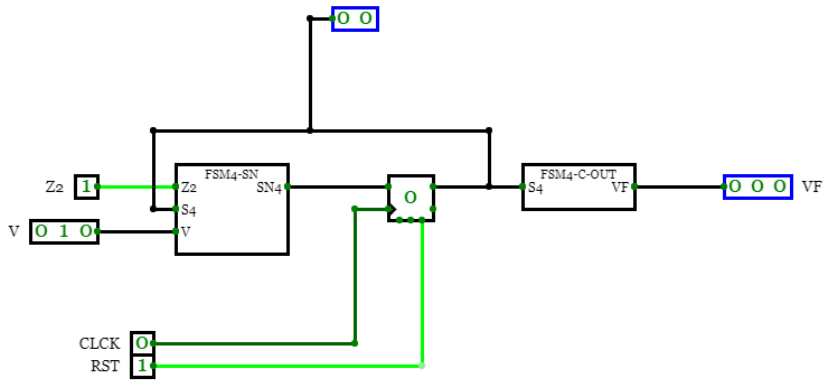
```
Minimized:
SN1 = S1 S0' Z' + S1' S0' V2' V1 V0' ;
SN0 = S1' S0 Z' + S1' S0' V2' V1' V0 ;
```

Salidas:

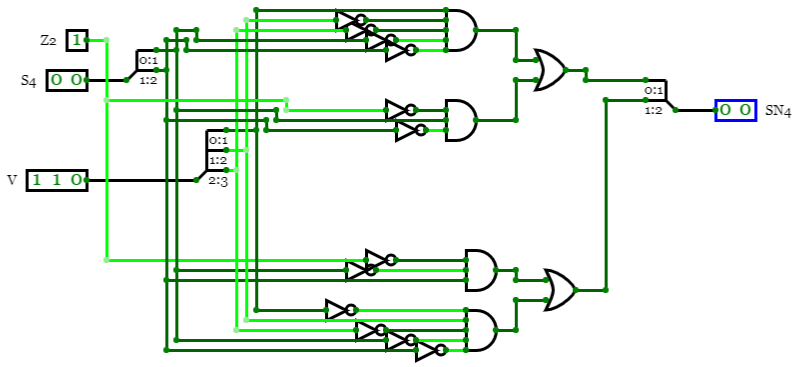
```
Minimized:
VF2 = 0;
VF1 = S1 S0';
VF0 = S1' S0;
```

Implementación en circuitverse:

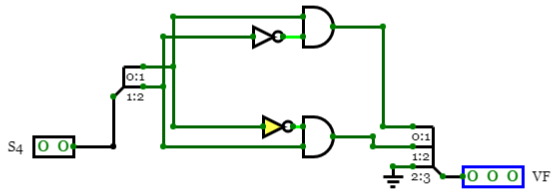
FSM:



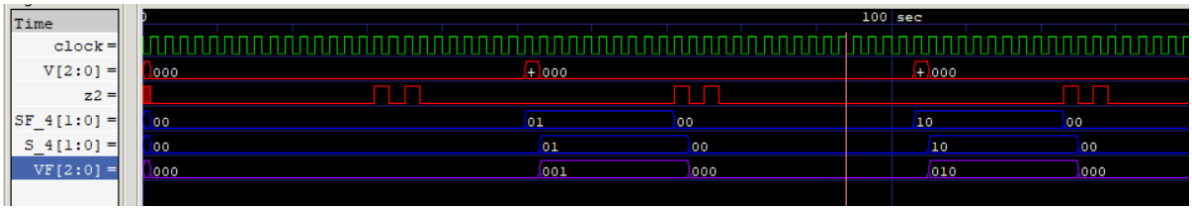
Estados:



Salidas:



Diagramas de timing e implementación en verilog:

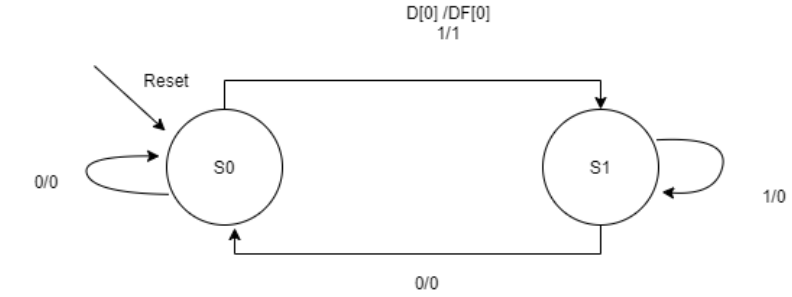


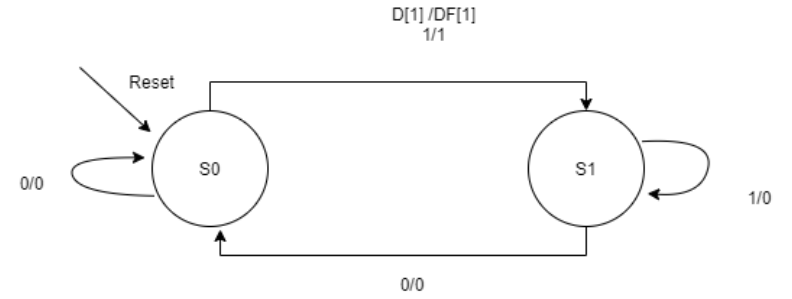
En el diagrama se observa que al momento en el que V (vuelto) tiene un pulso, la maquina cambia a un estado cuya salida es igual al vuelto ingresado, luego se mantiene hasta que z2 se coloque en 1 y luego regresa al estado inicial cuya salida 0 quedando a la espera de que otro valor sea ingresado.

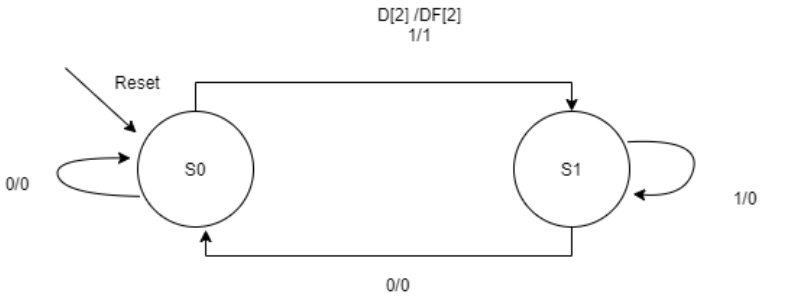
FSM debounce – Anti-rebote en los sensores de pago

Esta maquina consiste en garantizar que cuando el cliente ingrese su billete, la máquina de vuelto realice la resta una sola vez con ese billete, ya que, si no se tuviera esta máquina, al ingresar un billete, la entrada permanecería activa y el débito se realizaría varias veces con un solo billete.

Diagrama de estados:







Codificación de entradas, salidas y estados:

Dinero = D = DF
Q5 = 001
Q10 = 010
Q20 = 100
Estados
S0
S1

Tablas codificadas:

Estados y salidas:

S	V		SN	VF
0	0		0	0
0	1		1	1
1	1		1	0
1	0		0	0

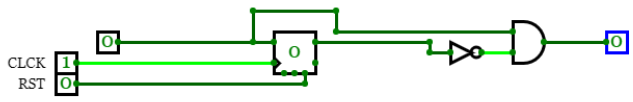
Ecuaciones booleanas:

Estados y salidas:

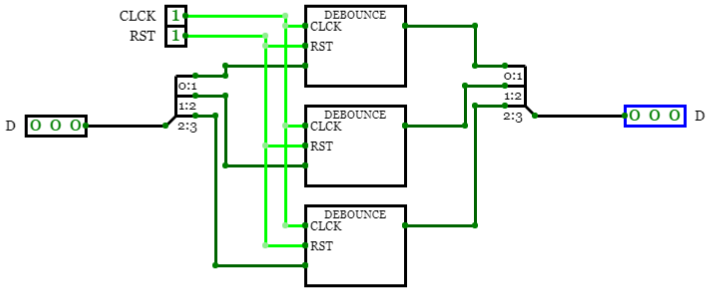
```
Entered by truthtable:
SN = S' V + S V;
VF = S' V;
```

Implementación en verilog:

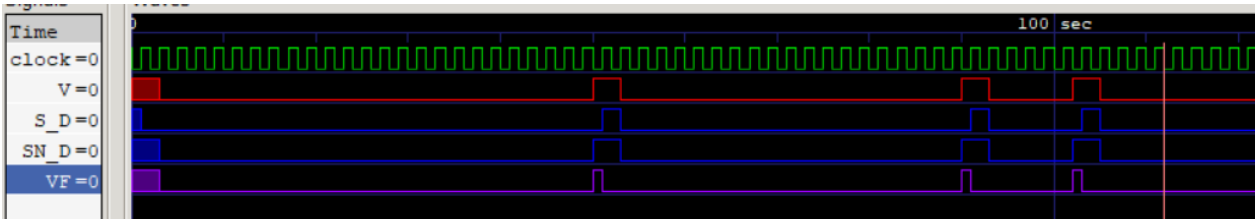
FSM de 1 bit:



FSM:



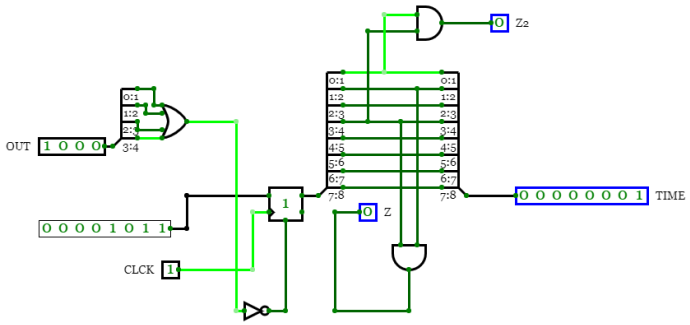
Diagramas de timing e implementación en verilog:



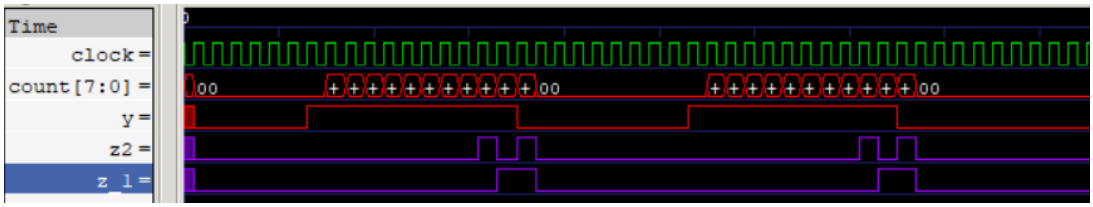
El diagrama que se observa corresponde a una FSM antirebote de 1 bit, en ella se observa que cuando la entrada tiene un pulso inmediatamente cambia de estado a una salida 1, si este pulso continúa cambia de estado a un estado cuya salida es 0 y se mantiene en este estado hasta que el pulso vuelve a ser 0 para regresar al estado inicial. Es importante observar que la salida se actualiza al mismo tiempo que la entrada, lo cual sucede debido a que esta es una máquina de mealy.

Timer

Implementación en circuitverse:



Diagramas de timing e implementación en verilog:



En el diagrama se puede observar que el timer no empieza a contar hasta que algún bit de OUT sea distinto de 0. Cuando Y se coloca en 1, COUNT comienza a incrementarse, cuando los bits 3 y 0 de count se encienden al mismo tiempo z2 se pone en 1 y cuando los bits 3 y 1 de COUNT se encienden al mismo tiempo, z_1 se pone en 1 y finalmente cuando COUNT llega al valor máximo asignado se reinicia y espera a que Y se vuelva a colocar en 1 para comenzar a contar.

Tabla de entradas y salidas obtenida con la simulación en verilog:

```
clk  P   D | OUT  VF
-----------|---------
1 xxxx xxx | 0000 000
0 xxxx xxx | 0000 000
1 0000 000 | 0000 000
0 0000 000 | 0000 000
1 0000 000 | 0000 000
0 0001 000 | 0000 000
1 0001 000 | 0000 000
0 0001 000 | 0000 000
1 0001 000 | 0000 000
0 0000 001 | 0000 000
1 0000 001 | 0000 000
0 0000 001 | 0000 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0001 000
0 0000 000 | 0001 000
1 0000 000 | 0000 000
0 0000 000 | 0000 000
1 0000 000 | 0000 000
0 0000 000 | 0000 000
1 0000 000 | 0000 000
0 0000 000 | 0000 000
1 0000 000 | 0000 000
0 0000 000 | 0000 000
1 0000 000 | 0000 000
```

Link del repositorio en github:

https://github.com/gil19443/Digital_1.git