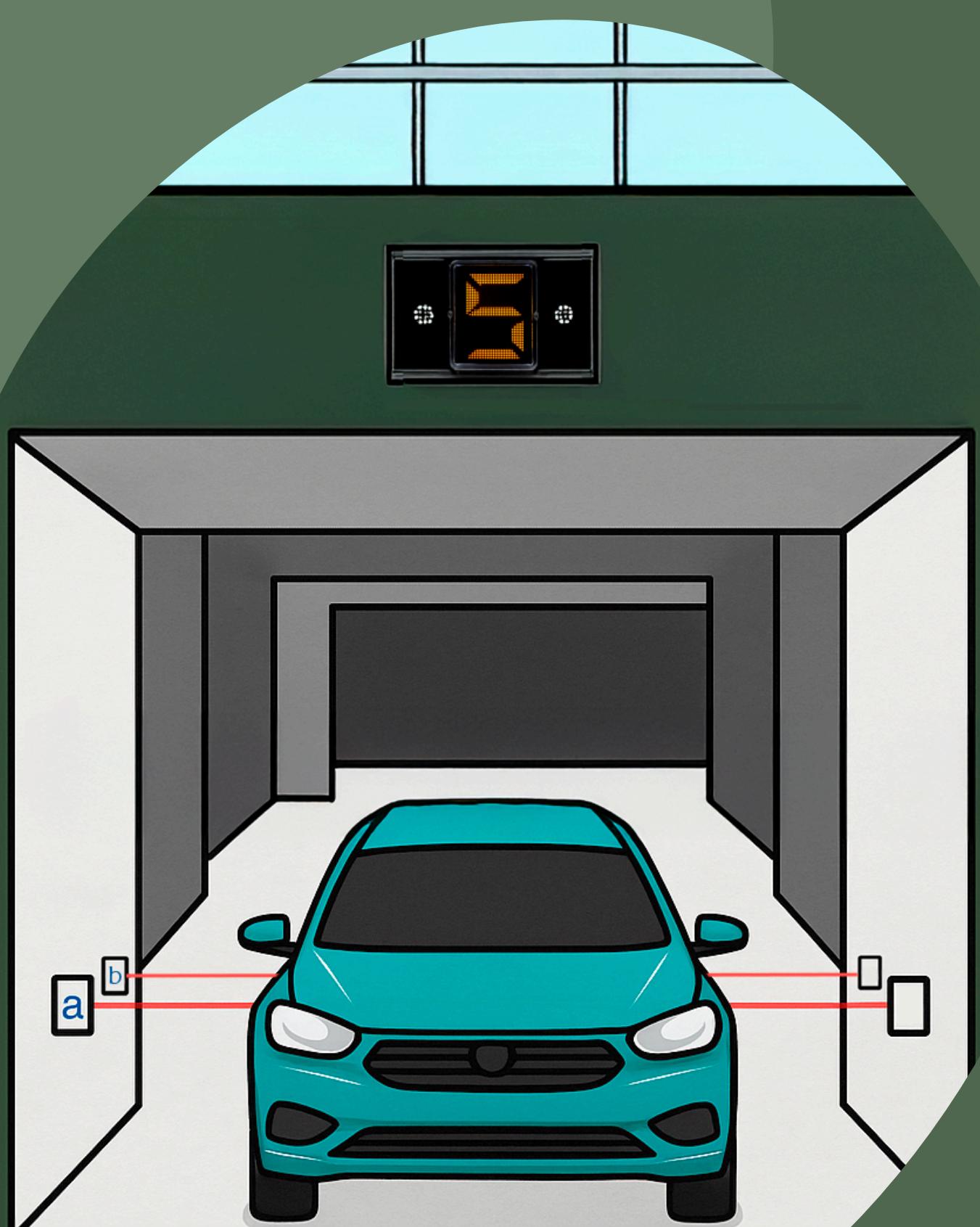


# Trabajo Final Integrador

Electrónica Digital

# Contador de ocupación de estacionamiento



# Solución al enunciado

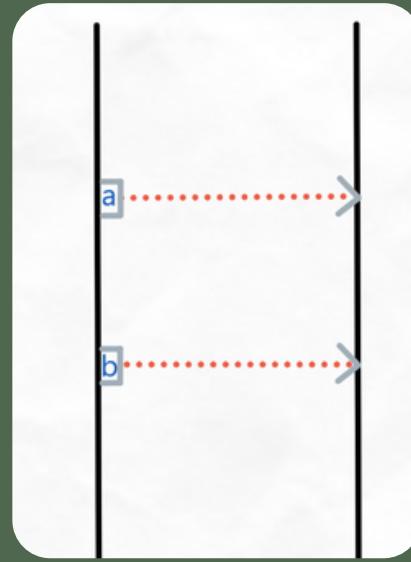
## Partes de la solución

- FSM para detectar ingreso y egreso de autos.
- FSM para un contador ascendente y descendente.
- Visualización del contador en los leds de la FPGA.

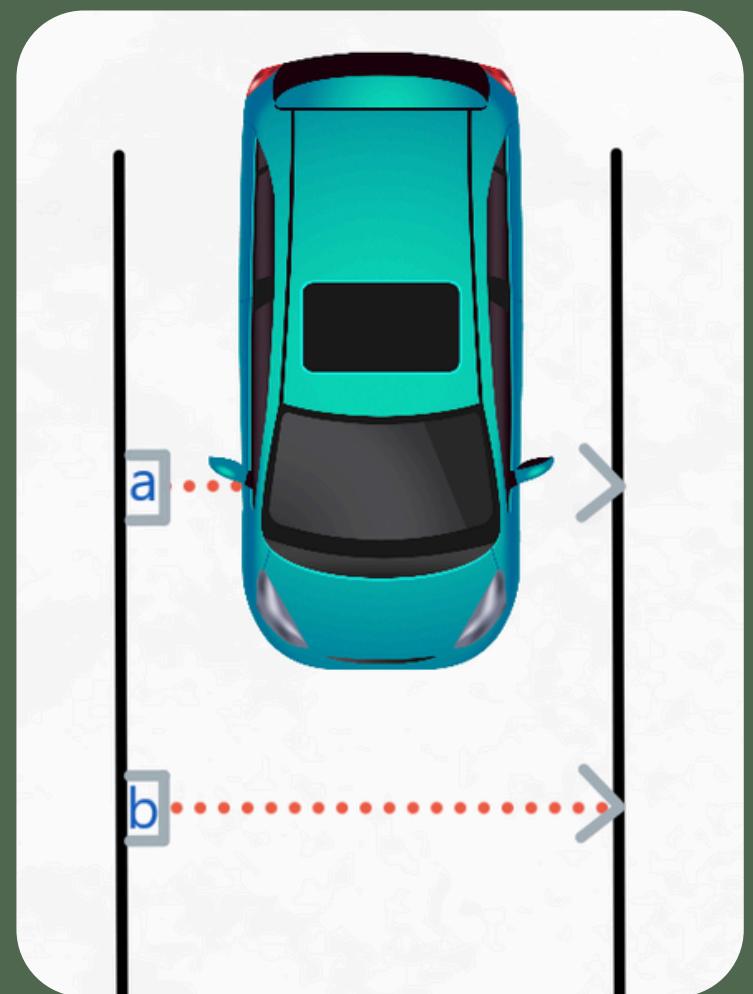
# Diagrama de estados

FSM - CONTROL DE ENTRADA Y SALIDA  
DEL ESTACIONAMIENTO

# Estados de los sensores a y b (ingreso)

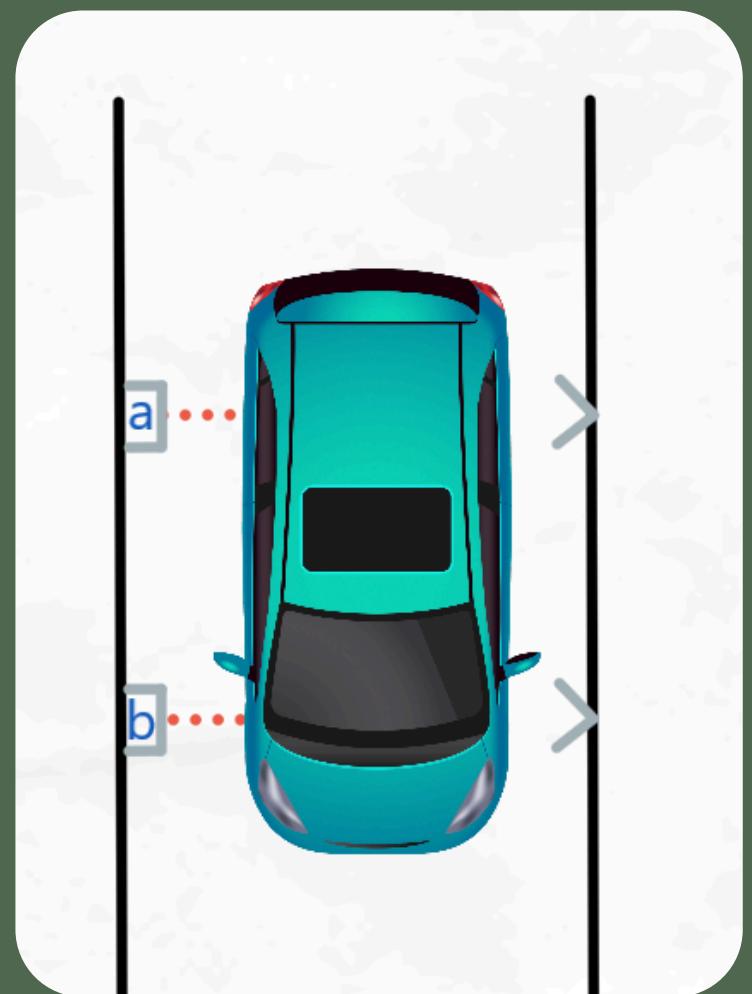


Estado 0  
Ambos sensores  
'a'y 'b' desbloqueados



Estado 1

Se bloquea el sensor 'a'  
primero



Estado 2

Se bloquean ambos  
sensores 'a' y 'b'



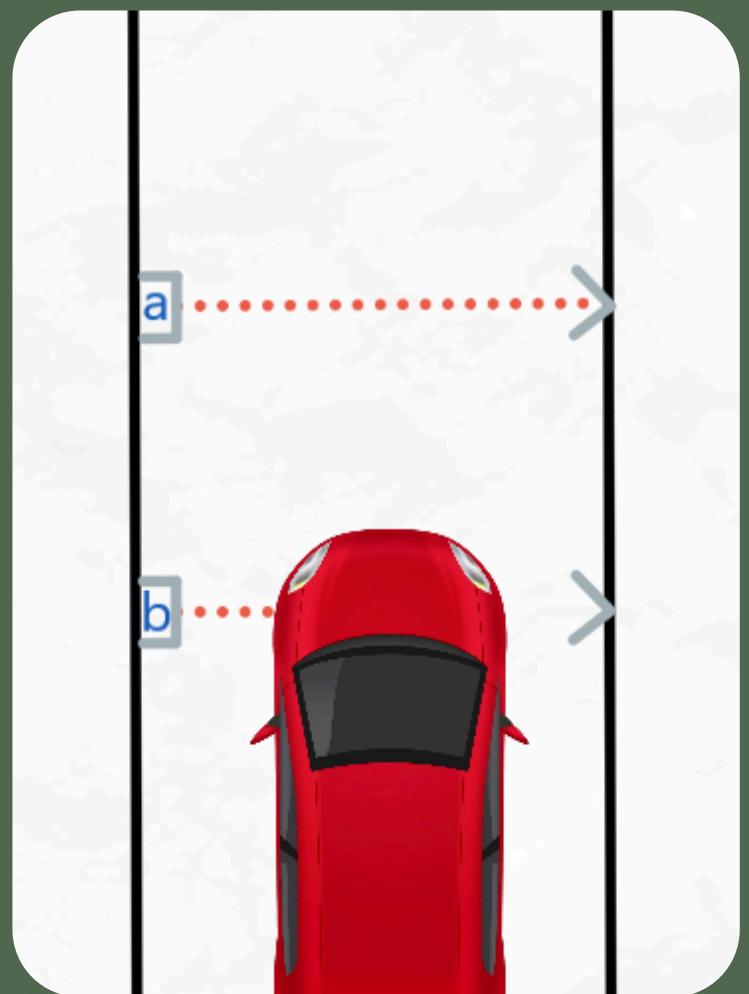
Estado 3

Se desbloquea el sensor 'a'

# Estados de los sensores a y b (Egreso)

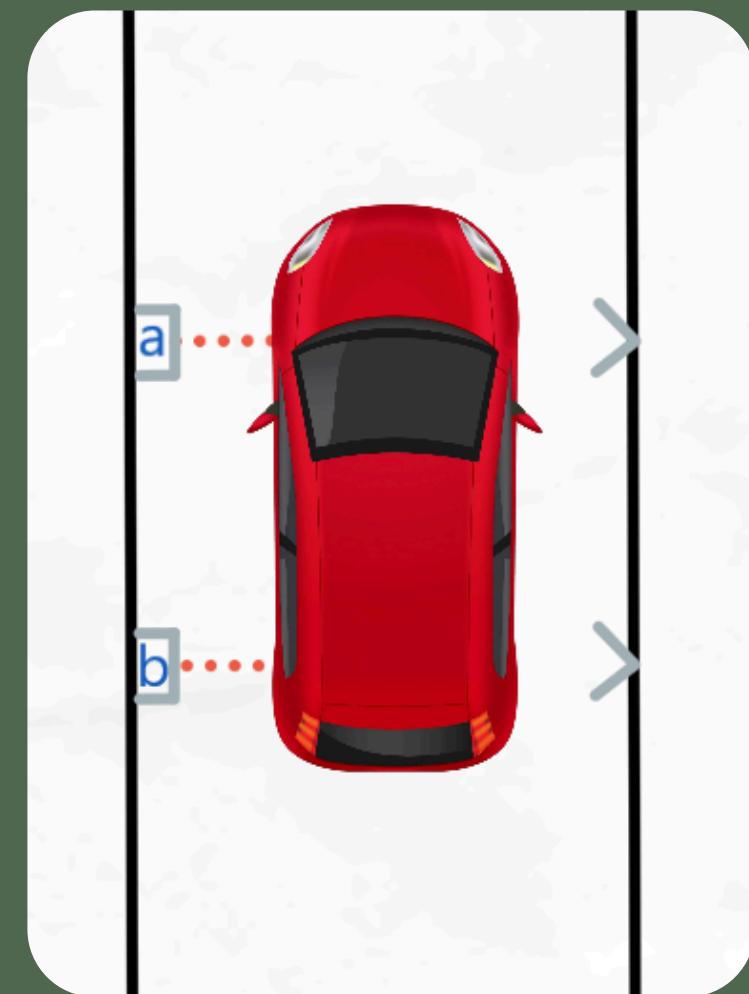


Estado 0  
Ambos sensores  
'a'y 'b' desbloqueados



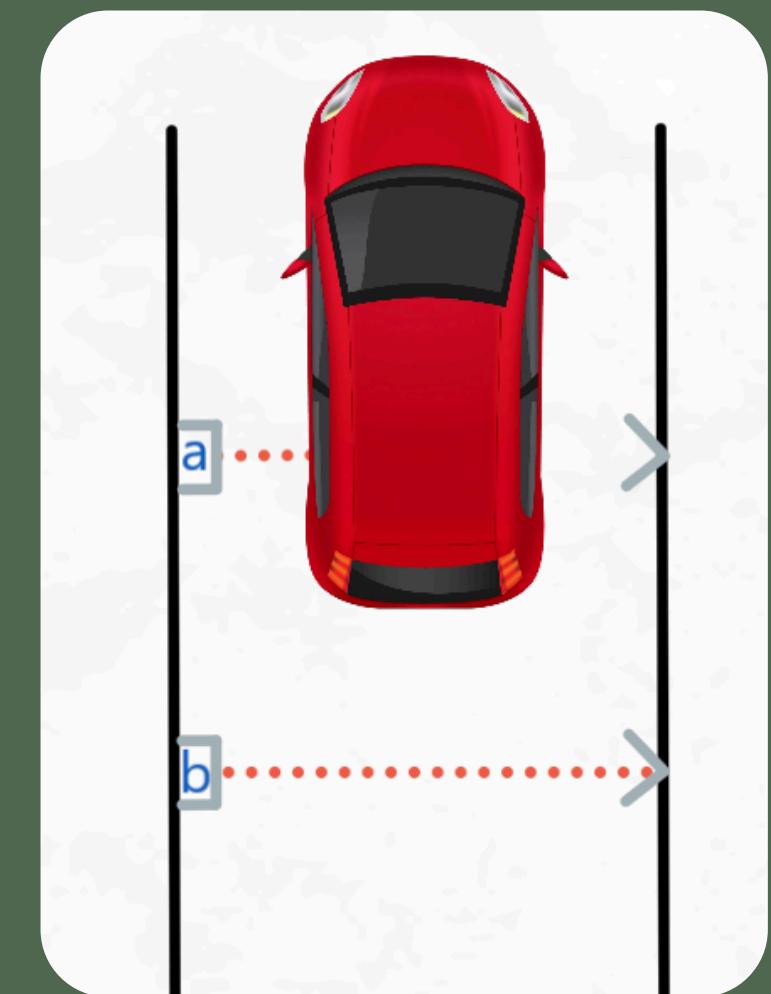
Estado 4

Se bloquea el sensor 'b'  
primero



Estado 5

Se bloquean ambos  
sensores 'a' y 'b'

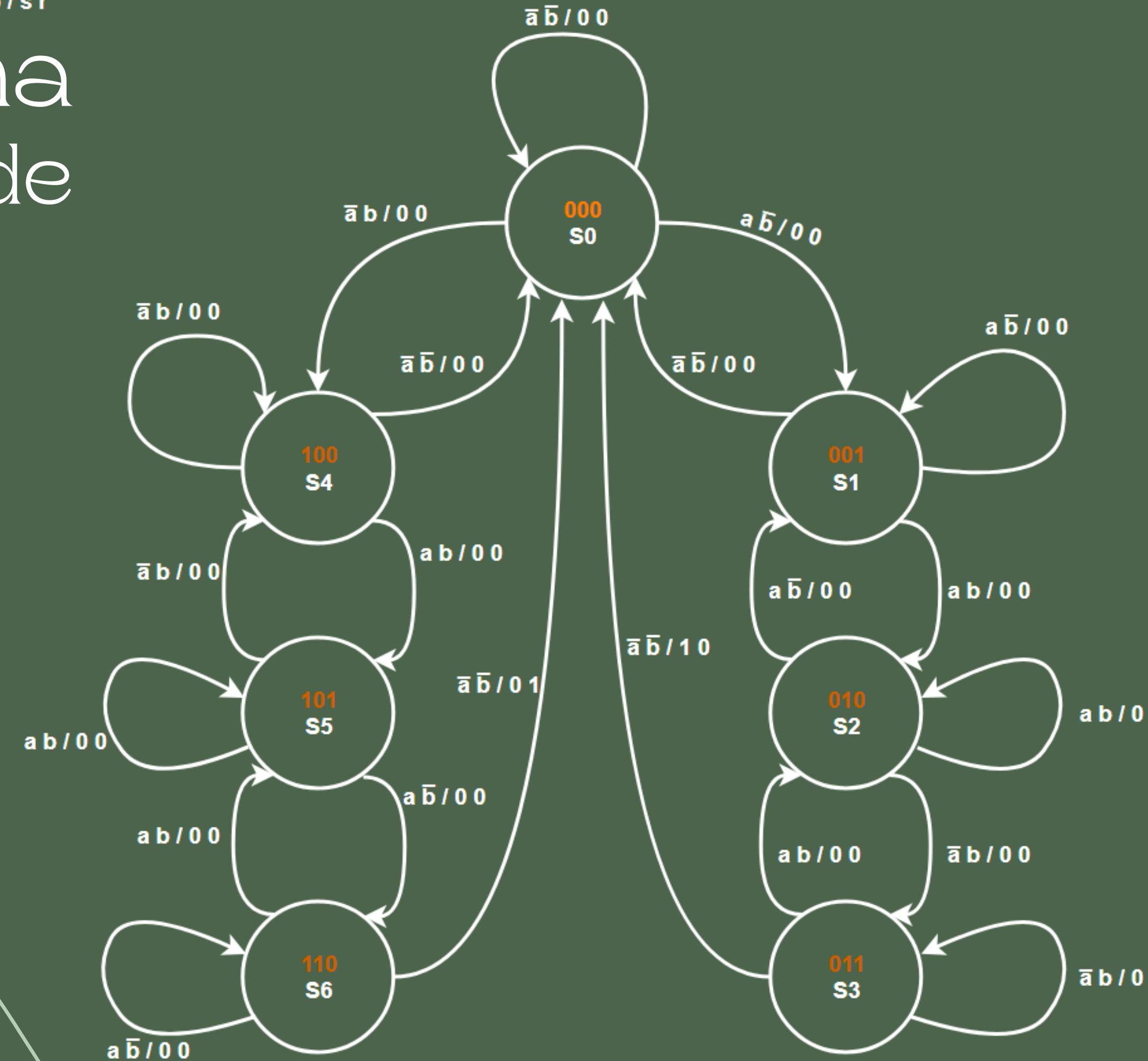


Estado 6

Se desbloquea el sensor 'b'

# Diagrama Máquina de Mealy

Entradas: a y b (sensores)  
Salidas: s y r ( indicadores de ingreso y egreso de autos)  
a b / s r



# Tabla de transición de estados y mapas de Karnaugh

FSM - CONTROL DE ENTRADA Y SALIDA  
DEL ESTACIONAMIENTO

# Tabla de transición

Entradas	Estados Actuales	Estados Siguientes	Salidas	Entradas Flip-Flop-D
a b	Q2 Q1 Q0 (t)	Q2 Q1 Q0 (t+1)	s r	D2 D1 D0
0 0	0 0 0 S0	0 0 0 S0	0 0	0 0 0
0 0	0 0 1 S1	0 0 0 S0	0 0	0 0 0
0 0	0 1 0 S2	X X X	X X	X X X
0 0	0 1 1 S3	0 0 0 S0	1 0	0 0 0
0 0	1 0 0 S4	0 0 0 S0	0 0	0 0 0
0 0	1 0 1 S5	X X X	X X	X X X
0 0	1 1 0 S6	0 0 0 S0	0 1	0 0 0
0 1	0 0 0 S0	1 0 0 S4	0 0	1 0 0
0 1	0 0 1 S1	X X X	X X	X X X
0 1	0 1 0 S2	0 1 1 S3	0 0	0 1 1
0 1	0 1 1 S3	0 1 1 S3	0 0	0 1 1
0 1	1 0 0 S4	1 0 0 S4	0 0	1 0 0
0 1	1 0 1 S5	1 0 0 S4	0 0	1 0 0
0 1	1 1 0 S6	X X X	X X	X X X
1 0	0 0 0 S0	0 0 1 S1	0 0	0 0 1
1 0	0 0 1 S1	0 0 1 S1	0 0	0 0 1
1 0	0 1 0 S2	0 0 1 S1	0 0	0 0 1
1 0	0 1 1 S3	X X X	X X	X X X
1 0	1 0 0 S4	X X X	X X	X X X
1 0	1 0 1 S5	1 1 0 S6	0 0	1 1 0
1 0	1 1 0 S6	1 1 0 S6	0 0	1 1 0
1 1	0 0 0 S0	X X X	X X	X X X
1 1	0 0 1 S1	0 1 0 S2	0 0	0 1 0
1 1	0 1 0 S2	0 1 0 S2	0 0	0 1 0
1 1	0 1 1 S3	0 1 0 S2	0 0	0 1 0
1 1	1 0 0 S4	1 0 1 S5	0 0	1 0 1
1 1	1 0 1 S5	1 0 1 S5	0 0	1 0 1
1 1	1 1 0 S6	1 0 1 S5	0 0	1 0 1

Tabla de excitación

Q(t)	Q(t+1)	D
0 → 0	0	
0 → 1	1	
1 → 1	1	
1 → 0	0	



"Se marcan con X X X aquellas transiciones que representan secuencias imposibles de detectar físicamente con el sistema de sensores, como por ejemplo el cambio de a=1, b=0 a a=0, b=1 sin pasar por a=1, b=1".

$a = 0$

D2		Q1 Q0		
b Q2	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1	1	1		
1 0	1			

$a = 1$

D2		Q1 Q0		
b Q2	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1	1	1		
1 0		1		

$$D_2 = b.Q_2.\overline{Q_1} + \bar{a}.b.\overline{Q_1}.\overline{Q_0} + a.Q_2.Q_1.\overline{Q_0} + a.Q_2.\overline{Q_1}.Q_0$$

$a = 0$

D1		Q1 Q0		
b Q2	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1				
1 0		1	1	

$a = 1$

D1		Q1 Q0		
b Q2	0 0	0 1	1 1	1 0
0 0				
0 1		1		1
1 1				
1 0		1	1	1

$$D_1 = b.\overline{Q_2}.Q_1 + a.b.\overline{Q_2}.Q_0 + a.\bar{b}.Q_2.\overline{Q_1}.Q_0 + a.\bar{b}.Q_2.Q_1.\overline{Q_0}$$

$a = 0$

D0		Q1 Q0		
b Q2	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1				
1 0		1	1	

$a = 1$

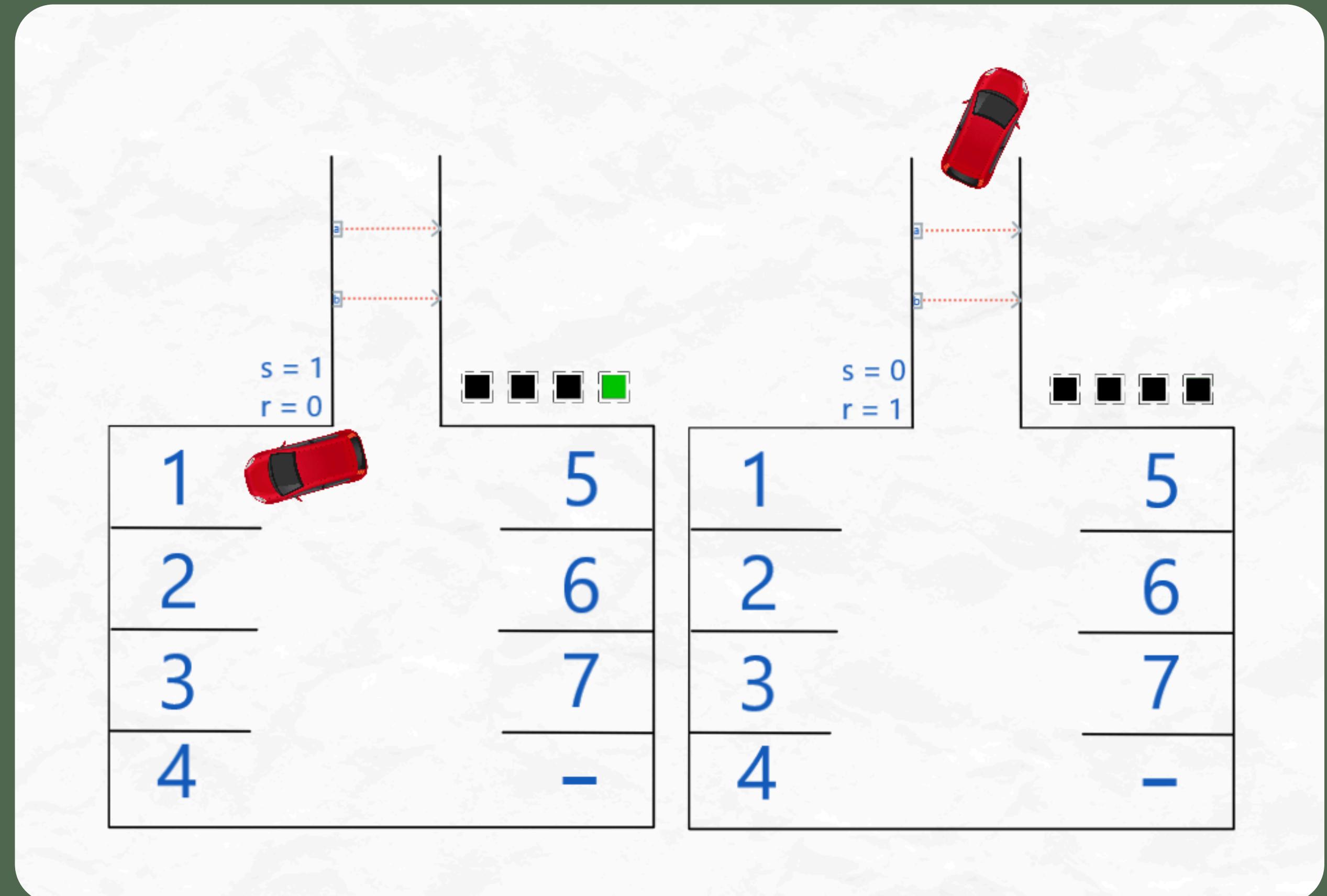
D0		Q1 Q0		
b Q2	0 0	0 1	1 1	1 0
0 0		1	1	
0 1				
1 1		1	1	
1 0				1

$$D_0 = \bar{a}.b.\overline{Q_2}.Q_1 + a.\bar{b}.\overline{Q_2}.\overline{Q_1} + a.b.Q_2.\overline{Q_1} + a.\bar{b}.Q_2.\overline{Q_0} + a.b.Q_2.\overline{Q_0}$$

# Diagrama de estados

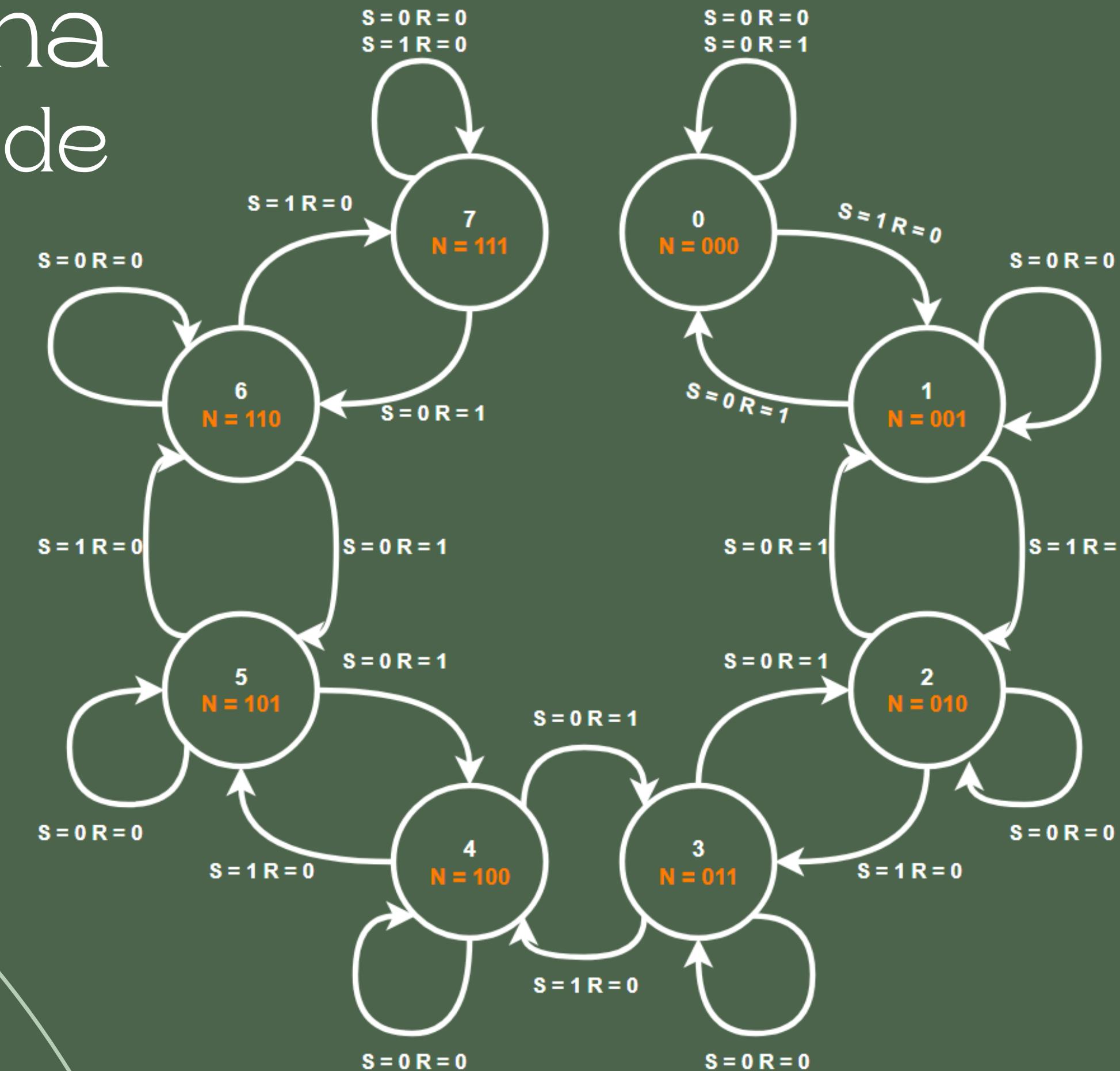
FSM - CONTADOR ASCENDENTE Y  
DESCENDENTE 0-7

# Comportamiento del contador por las entradas 's' y 'r'



Entradas: S Y R (Ingreso y egreso de autos)  
Salidas: [2:0] N (Indicador de cuantos autos hay en el estacionamiento)

# Diagrama Máquina de Moore



# Tabla de transición de estados y mapas de Karnaugh

FSM - CONTADOR ASCENDENTE Y  
DESCENDENTE 0-7

# Tabla de transición

Entradas	Estados Actuales			Salidas	Entradas Flip-Flops-D	
	S	R	Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub> (t)	Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub> (t+1)		
0 0	0 0 0	S0	0 0 0	S0	0 0 0	0 0 0
0 0	0 0 1	S1	0 0 1	S1	0 0 1	0 0 1
0 0	0 1 0	S2	0 1 0	S2	0 1 0	0 1 0
0 0	0 1 1	S3	0 1 1	S3	0 1 1	0 1 1
0 0	1 0 0	S4	1 0 0	S4	1 0 0	1 0 0
0 0	1 0 1	S5	1 0 1	S5	1 0 1	1 0 1
0 0	1 1 0	S6	1 1 0	S6	1 1 0	1 1 0
0 0	1 1 1	S7	1 1 1	S7	1 1 1	1 1 1
0 1	0 0 0	S0	0 0 0	S0	0 0 0	0 0 0
0 1	0 0 1	S1	0 0 0	S0	0 0 1	0 0 0
0 1	0 1 0	S2	0 0 1	S1	0 1 0	0 0 1
0 1	0 1 1	S3	0 1 0	S2	0 1 1	0 1 0
0 1	1 0 0	S4	0 1 1	S3	1 0 0	0 1 1
0 1	1 0 1	S5	1 0 0	S4	1 0 1	1 0 0
0 1	1 1 0	S6	1 0 1	S5	1 1 0	1 0 1
0 1	1 1 1	S7	1 1 0	S6	1 1 1	1 1 0
1 0	0 0 0	S0	0 0 1	S1	0 0 0	0 0 1
1 0	0 0 1	S1	0 1 0	S2	0 0 1	0 1 0
1 0	0 1 0	S2	0 1 1	S3	0 1 0	0 1 1
1 0	0 1 1	S3	1 0 0	S4	0 1 1	1 0 0
1 0	1 0 0	S4	1 0 1	S5	1 0 0	1 0 1
1 0	1 0 1	S5	1 1 0	S6	1 0 1	1 1 0
1 0	1 1 0	S6	1 1 1	S7	1 1 0	1 1 1
1 0	1 1 1	S7	1 1 1	S7	1 1 1	1 1 1

Tabla de excitación

Q(t)	Q(t+1)	D
0 → 0	0	
0 → 1	1	
1 → 1	1	
1 → 0	0	

$S = 0$ 

D2		Q1 Q0		
R Q2	0 0	0 1	1 1	1 0
0 0				
0 1	1	1	1	1
1 1	1	1	1	1
1 0				

 $S = 0$ 

D2		Q1 Q0		
R Q2	0 0	0 1	1 1	1 0
0 0				
0 1	1	1	1	1
1 1				
1 0				

 $S = 1$ 

D1		Q1 Q0		
R Q2	0 0	0 1	1 1	1 0
0 0				
0 1		1	1	1
1 1	1	1	1	1
1 0				

 $S = 0$ 

D1		Q1 Q0		
R Q2	0 0	0 1	1 1	1 0
0 0				
0 1		1	1	1
1 1				
1 0				

 $S = 1$ 

D0		Q1 Q0		
R Q2	0 0	0 1	1 1	1 0
0 0				
0 1	1	1	1	1
1 1	1	1	1	1
1 0				



D0		Q1 Q0		
R Q2	0 0	0 1	1 1	1 0
0 0				
0 1	1	1	1	1
1 1				
1 0				



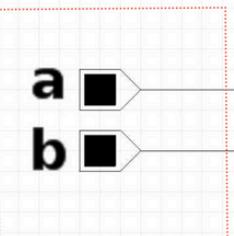
$$D_2 = \overline{R} \cdot Q_2 + \overline{S} \cdot Q_2 \cdot Q_0 + \overline{S} \cdot Q_2 \cdot Q_1 + S \cdot \overline{R} \cdot Q_1 \cdot Q_0$$

$$D_1 = \overline{R} \cdot Q_1 \cdot \overline{Q}_0 + \overline{S} \cdot Q_1 \cdot Q_0 + \overline{S} \cdot R \cdot Q_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0 + S \cdot \overline{R} \cdot \overline{Q}_1 \cdot Q_0 + S \cdot \overline{R} \cdot Q_2 \cdot Q_0$$

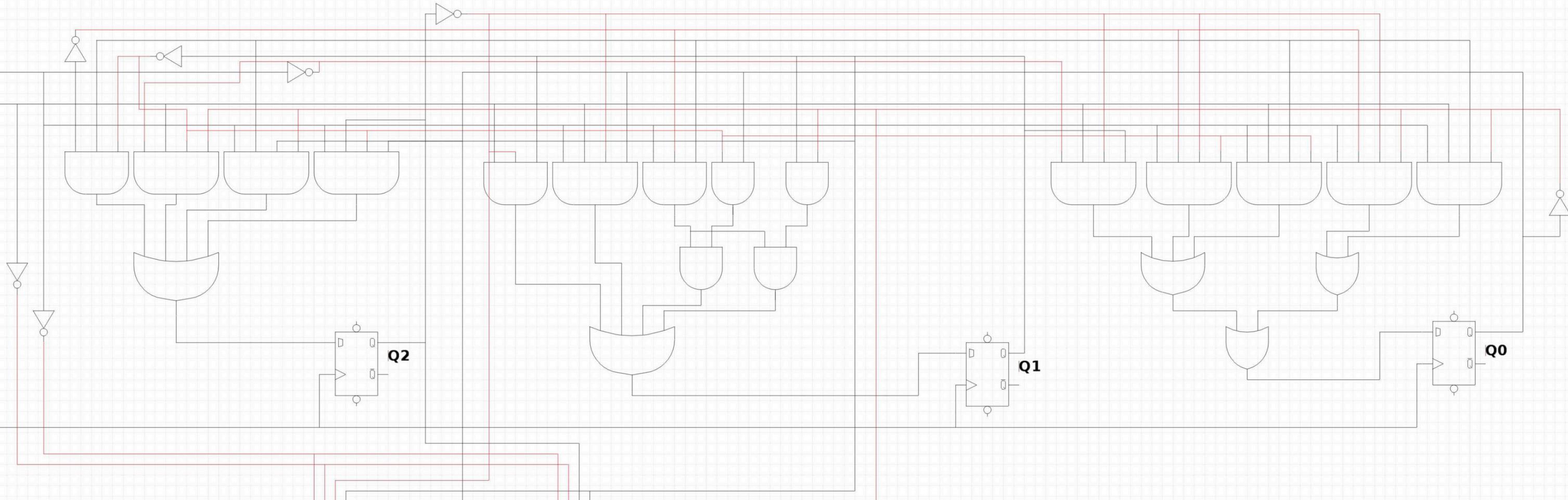
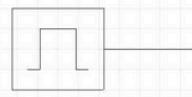
$$D_0 = \overline{S} \cdot \overline{R} \cdot Q_0 + \overline{S} \cdot R \cdot Q_1 \cdot \overline{Q}_0 + \overline{S} \cdot R \cdot Q_2 \cdot \overline{Q}_0 + S \cdot \overline{R} \cdot Q_2 \cdot Q_1 + S \cdot \overline{R} \cdot \overline{Q}_0$$



Implementación  
de las FSM  
utilizando  
circuitos lógicos

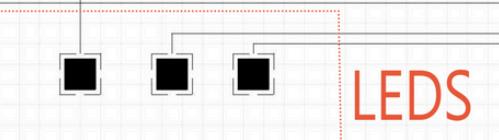
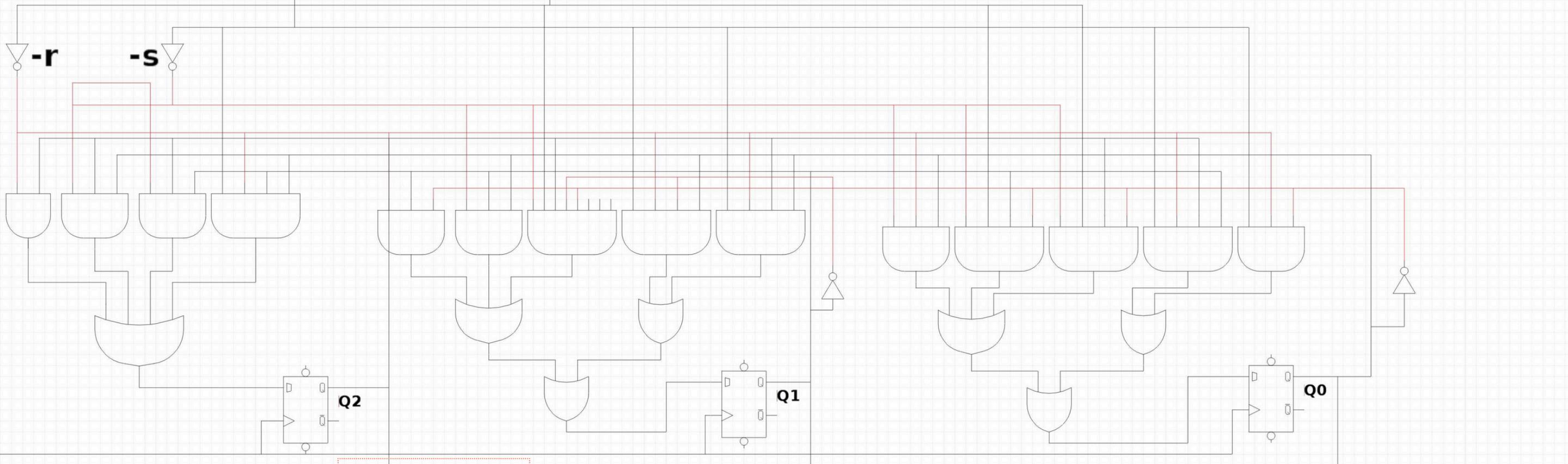


**PULSADORES**



**s**      **r**

**-r**      **-s**

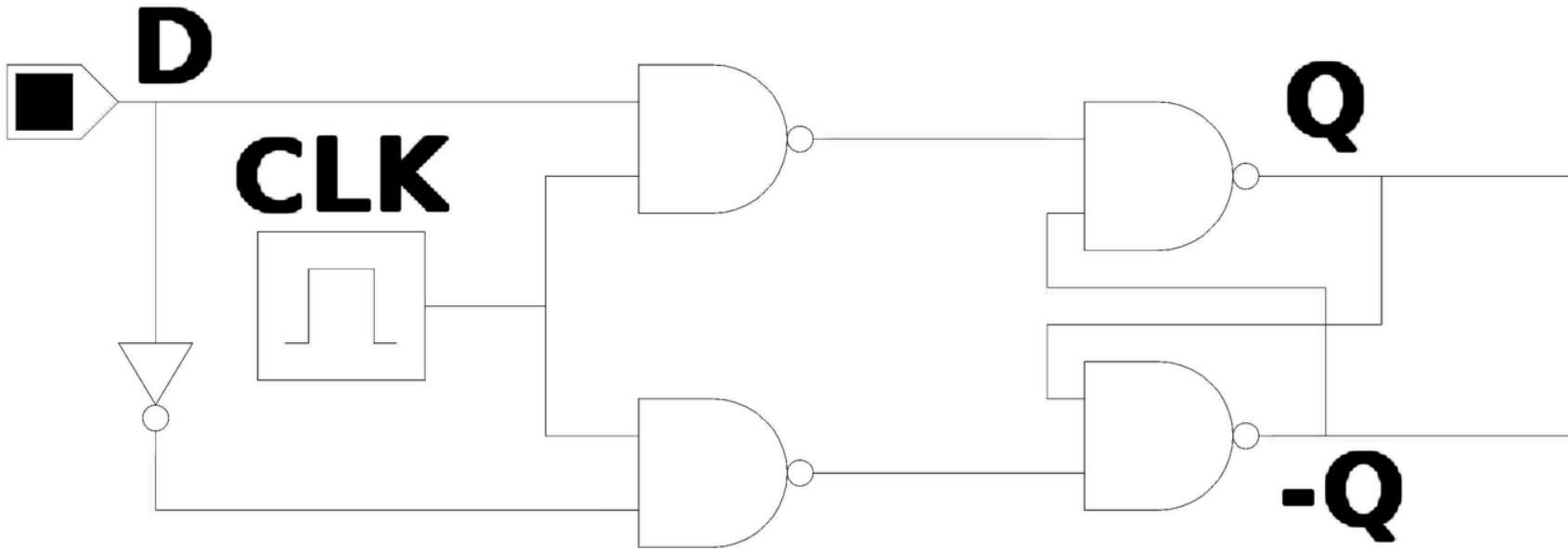


**LEDS**



Solución del  
problema en  
Verilog

# Flip-Flop D



Implementación con  
compuertas lógicas

```
1 // Flip-Flop D activado por flanco de subida
2 module FF_D(
3     input wire D,
4     input wire clk,
5     output reg Q = 0,
6     output wire Qn
7 );
8     always @ (posedge clk) begin
9         Q <= D;
10    end
11    assign Qn = ~Q;
12 endmodule
```

Implementación en Verilog

# **FSM**

## **CONTROL DE ENTRADA Y SALIDA DEL ESTACIONAMIENTO**



```
1 // Módulo de una máquina de estados finitos (FSM) que controla la entrada y salida de autos en un estacionamiento.
2 module FSM_control_estacionamiento(
3     input wire clk, // Reloj
4     input wire a, // Entrada que representa al 1er sensor del estacionamiento
5     input wire b, // Entrada que representa al 2do sensor del estacionamiento
6     output wire S, // Salida que representa el ingreso de un auto al estacionamiento
7     output wire R // Salida que represeta la salida de un auto del estacionamiento
8 );
9
10    // Entradas para los flip-flops tipo D
11    wire D2, D1, D0;
12
13    // Salidas de los flip-flops tipo D
14    wire Q2, Q1, Q0;
15
16    // Implementación de la lógica combinacional para las entradas D de los flip-flops
17    assign D2 = (b & Q2 & ~Q1) | (~a & b & ~Q1 & ~Q0) |
18        (a & Q2 & Q1 & ~Q0) | (a & Q2 & ~Q1 & Q0);
19
20    assign D1 = (b & ~Q2 & Q1) | (a & b & ~Q2 & Q0) |
21        (a & ~b & Q2 & ~Q1 & Q0) | (a & ~b & Q2 & Q1 & ~Q0);
22
23    assign D0 = (~a & b & ~Q2 & Q1) | (a & ~b & ~Q2 & ~Q1) |
24        (a & b & Q2 & ~Q1) | (a & ~b & ~Q2 & ~Q0) |
25        (a & b & Q2 & ~Q0);
26
27    // Instanciación de los flip-flops tipo D
28    FF_D ff2 (.D(D2), .clk(clk), .Q(Q2), .Qn());
29    FF_D ff1 (.D(D1), .clk(clk), .Q(Q1), .Qn());
30    FF_D ff0 (.D(D0), .clk(clk), .Q(Q0), .Qn());
31
32    // Asignación de las salidas S y R (ingreso y salida de autos)
33    assign S = ~a & ~b & ~Q2 & Q1 & Q0;
34    assign R = ~a & ~b & Q2 & Q1 & ~Q0;
35
36 endmodule
```

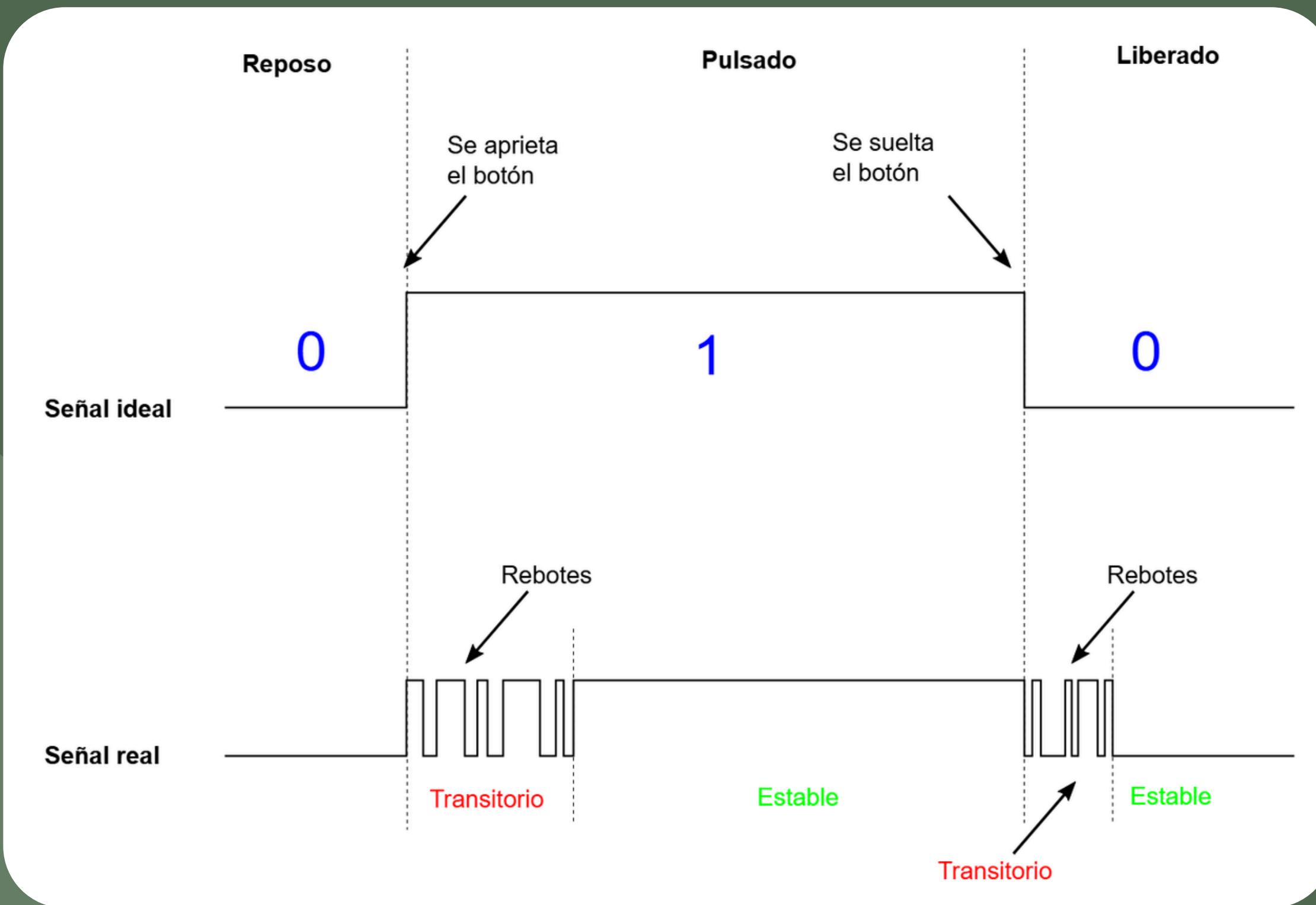
# **FSM CONTADOR ASCENDENTE Y DESCENDENTE**



```
1 // Contador ascendente y descendente de 3 bits con flip-flops tipo D
2 module contador_ascendente_descendente(
3     input wire clk, // Reloj
4     input wire r, // Entrada que representa la salida de un auto del estacionamiento
5     input wire s, // Entrada que representa el ingreso de un auto al estacionamiento
6     output wire [2:0] leds // Salida de 3 bits que indica el estado del contador
7 );
8
9     // Entradas para los flip-flops tipo D
10    wire D2, D1, D0;
11
12    // Salidas de los flip-flops tipo D
13    wire Q2, Q1, Q0;
14
15    // Implementación de la lógica combinacional para las entradas D de los flip-flops
16    assign D2 = (~r & Q2) | (~s & Q2 & Q0) | (~s & Q2 & Q1) | (s & ~r & Q1 & Q0);
17    assign D1 = (~r & Q1 & ~Q0) | (~s & Q1 & Q0) | (~s & r & Q2 & ~Q1 & ~Q0) |
18                  (s & ~r & ~Q1 & Q0) | (s & ~r & Q2 & Q0);
19    assign D0 = (~s & ~r & Q0) | (~s & r & Q1 & ~Q0) | (~s & r & Q2 & ~Q0) |
20                  (s & ~r & Q2 & Q1) | (s & ~r & ~Q0);
21
22    // Instanciación de los flip-flops tipo D
23    FF_D ffd_2(.D(D2), .clk(clk), .Q(Q2), .Qn());
24    FF_D ffd_1(.D(D1), .clk(clk), .Q(Q1), .Qn());
25    FF_D ffd_0(.D(D0), .clk(clk), .Q(Q0), .Qn());
26
27    // Asignación de las salidas del contador a el registro que representa los LEDs
28    // que indican el estado del estacionamiento
29    assign leds = {Q2, Q1, Q0};
30
31 endmodule
```

# Implementación Antirebote de los pulsadores

# Señal ideal vs Señal real





```
1 // Descripción: Módulo para eliminar el rebote de un botón
2 module antirebote(
3     input wire BTN, // Entrada del botón que se desea estabilizar
4     input wire clk, // Reloj del sistema
5     output wire BTN_estable // Salida del botón estabilizado
6 );
7
8 // Registros para almacenar los valores intermedios
9 reg btn_aux1 = 1'b0; // Guarda el valor más reciente del botón
10 reg btn_aux2 = 1'b0; // Guarda el valor estabilizado del botón
11
12 /*
13 Este contador de 17 bits se utiliza para esperar cierto tiempo antes de considerar que
14 la señal del botón es estable. La idea es: si BTN no cambia durante un tiempo suficientemente
15 largo (por ejemplo, unos milisegundos), entonces se considera que el valor actual es confiable.
16 */
17 reg [16:0] contador = 0; // Contador para el tiempo de estabilización
18
19 always @(posedge clk) begin
20
21     if (btn_aux1 ^ BTN == 1'b1)
22         begin
23             // Si hay un cambio en el botón, reiniciar el contador
24             contador <= 0;
25             btn_aux1 <= BTN; // Actualizar al valor mas reciente del botón
26         end
27     else if (contador[16] == 1'b0) // Se espera que el bit más alto del contador (contador[16]) llegue a 1, lo cual indica que ha pasado suficiente tiempo sin cambios.
28         contador <= contador + 1; // Incrementar el contador si no hay cambio
29     else
30         btn_aux2 <= btn_aux1; // Si el contador ha llegado al final, actualizar la salida estable
31
32 end
33
34 assign BTN_estable = btn_aux2; // Asignar el valor del segundo registro auxiliar a la salida estable
35
36 endmodule
```

# Resolución del TFI

```
1 module TFI(
2     input CLK, // Reloj de la edu-ciaa-fpga
3     input BTN1, // Entrada que repesenta al par de sensores 'a' del estacionamiento
4     input BTN4, // Entrada que repreesnta al par de sensores 'b' del estacionamiento
5     // Salidas para los LEDs que indican el estado del estacionamiento
6     output wire LED3,
7     output wire LED2,
8     output wire LED1,
9     output wire LED0
10 );
11
12     wire BTN1_estable; // Botón 1 estabilizado (sensor a)
13     // Instanciación del módulo debouncer para estabilizar la señal del botón 1
14     antirebote antirebote boton1(
15         .BTN(~BTN1),
16         .clk(CLK),
17         .BTN_estable(BTN1_estable)
18     );
19
20     wire BTN4_estable; // Botón 4 estabilizado (sensor b)
21     // Instanciación del módulo debouncer para estabilizar la señal del botón 4
22     antirebote antirebote boton4(
23         .BTN(~BTN4),
24         .clk(CLK),
25         .BTN_estable(BTN4_estable)
26     );
27
28     wire s; // Señal de entrada para el contador ascendente/descendente (ingreso de auto)
29     wire r; // Señal de salida para el contador ascendente/descendente (salida de auto)
30
31     // Instanciación de la máquina de estados finitos (FSM) que controla la entrada y salida de autos
32     FSM_control_estacionamiento fsm(
33         .clk(CLK),
34         .a(BTN1_estable),
35         .b(BTN4_estable),
36         .S(s),
37         .R(r)
38     );
39
40     wire [2:0] leds; // Salidas del contador ascendente/descendente que indican el estado del estacionamiento
41
42     // Instanciación del contador ascendente/descendente que lleva el registro de autos en el estacionamiento
43     contador_ascendente_descendente contador(
44         .clk(CLK),
45         .r(r),
46         .s(s),
47         .leds(leds)
48     );
49
50     assign LED2 = leds[2];
51     assign LED1 = leds[1];
52     assign LED0 = leds[0];
```



```
1 // Parámetros para calcular el número de ciclos de 2 segundos
2 parameter CLK_FREQ = 12000000; // 12MHz, son 12.000.000 ciclos por segundo
3 parameter dos_segundos = CLK_FREQ * 2; // 2 segundos en ciclos de reloj
4
5 // Registro para controlar el parpadeo del LED3
6 // Con 25 bits puedo contar hasta 33,554,432 ciclos
7 reg [25:0] contador2 = 0; //Utilizado para contar los ciclos de reloj
8 reg parpadeo = 0;
9
10 always @(posedge CLK) begin
11     if (leds == 3'b111) begin
12         if (contador2 >= dos_segundos - 1) begin
13             contador2 <= 0;
14             parpadeo <= ~parpadeo;
15         end else begin
16             contador2 <= contador2 + 1;
17         end
18     end else begin // Nos aseguramos que el LED3 se apague y se reinicie el contador2
19         contador2 <= 0;
20         parpadeo <= 0;
21     end
22 end
23
24 // LED3 titila solo cuando leds == 3'b111
25 assign LED3 = (leds == 3'b111) ? parpadeo : 1'b0;
26
27 endmodule
```

Muchas  
gracias!

Godoy Matías Nahuel - Palacios Angel Luis