Informe de Laboratorio: Sistema de rutas criminales

Vicente Rodríguez Rogers 202273503-1
Nicolás Muñoz 202273641-0

Noviembre 2025

1. Introducción

En el presente informe, se documentará el desarrollo de un sistema de rastreo secuencial esencial para contrarrestar el nuevo modus operandi de la organización criminal. Tras el éxito del decodificador, los métodos han evolucionado y ahora se utilizan códigos de 4 bits para describir puntos de partida para rutas de escape dinámicas a través de una red de 16 nodos que convergen en un punto de extracción final. Para cumplir con la misión encomendada se realizará un circuito que utilice lógica combinacional y flip-flops tipo D, que permitan rastrear el movimiento de los criminales paso a paso y mostrar la ubicación en un display de 7 segmentos.

2. Análisis

2.1. Ubicación y estados

El problema requiere de diseñar un autómata de estados finitos que simule el recorrido del grafo, y para lograr esto primero hay que definir los estados del sistema (que en este caso serian las 16 ubicaciones clandestinas). Por lo que definimos una tabla indicando los estados y sus codificaciones;

Ubicación	**Decimal**	**código** ($Q_3Q_2Q_1Q_0$)
A	0	0000
b	1	0001
C	2	0010
d	3	0011
E	4	0100
F **(Extracción)**	5	0101
G	6	0110
H	7	0111
I	8	1000
J	9	1001
K	10	1010
L	11	1011
M	12	1100
N	13	1101
O	14	1110
P	15	1111

Cuadro 1: Tabla de códigos de estados.

2.2. Representación de ubicaciones

Hay que representar cada posición en un display según la ubicación en la que esta ubicada la banda criminal, es por eso que hacemos uso de un display que nos ayudara con esta representación cuya codificación viene dada por;

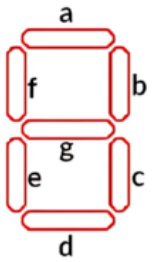


Figura 1: *
Representación de un display de 7 segmentos

Símbolo	**Segmentos Activos**	**abcdefg**
A	a,b,c,e,f,g	1110111
b	f,e,d,c,g	0011111
C	a,f,e,d	1001110
d	b,c,d,e,g	0111101
E	a,f,g,e,d	1001111
F	a,f,g,e	1000111
G	a,c,d,e,f	1011110
H	b,c,e,f	0110111
I	b,c	0110000
J	b,c,d,e	0111100
K	a,c,e,f,g	1010111
L	d,e,f	0001110
M	a,c,e,g	1010101
N	a,c,e	1010100
O	a,b,c,d,e,f	1111110
P	a,b,e,f,g	1110011

Figura 2: *
Tabla de Segmentos y Códigos Binarios

2.3. Diagrama de estados

Por ultimo, debemos crear un diagrama de estados que nos permita definir como se va a comportar nuestro circuito a la hora de hacerlo funcionar. Para esto diseñamos el siguente diagrama

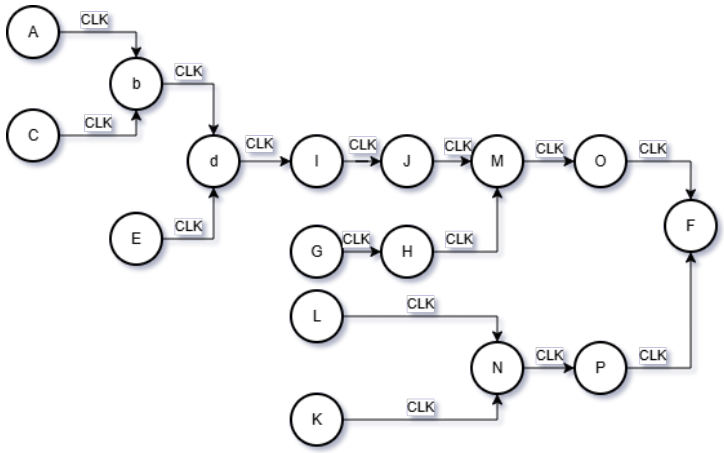


Figura 3: Diagrama de transición de estados

3. Desarrollo

3.1. Diseño, Tablas y minimización

- **Decodificador de entrada:** Este será un bloque que convierte la entrada ($A_3A_2A_1A_0$) en el estado inicial ($Q_3Q_2Q_1Q_0$)

    1. **Tabla:**

Entrada				Nodo	Salida (Inicial)			
$\mathbf{A_3}$	$\mathbf{A_2}$	$\mathbf{A_1}$	$\mathbf{A_0}$	-	$\mathbf{Q_3}$	$\mathbf{Q_2}$	$\mathbf{Q_1}$	$\mathbf{Q_0}$
0	0	0	0	A	0	0	0	0
0	0	0	1	B	0	0	0	1
0	0	1	0	C	0	0	1	0
0	0	1	1	D	0	0	1	1
0	1	0	0	E	0	1	0	0
0	1	0	1	F	0	1	0	1
0	1	1	0	G	0	1	1	0
0	1	1	1	H	0	1	1	1
1	0	0	0	I	1	0	0	0
1	0	0	1	J	1	0	0	1
1	0	1	0	K	1	0	1	0
1	0	1	1	L	1	0	1	1
1	1	0	0	M	1	1	0	0
1	1	0	1	N	1	1	0	1
1	1	1	0	O	1	1	1	0
1	1	1	1	P	1	1	1	1

Cuadro 2: Tabla de Verdad del Decodificador de Entrada (Inicialización)

2. **Funciones Booleanas Minimizadas:**

$$Q_3 = A_3$$

$$Q_2 = A_2$$

$$Q_1 = A_1$$

$$Q_0 = A_0$$

- **Lógica de transición de estado:** este bloque es el eje central de nuestro circuito, ya que es el encargado de definir el valor que tendrán los flip-flops D ($D_3D_2D_1D_0$) basándose en el estado actual ($Q_3Q_2Q_1Q_0$).

1. **Tabla de transiciones:**

Estado Actual (Q_{actual})		**Próximo Estado (D_{next})**	
Nodo	**$Q_3Q_2Q_1Q_0$**	**Nodo Siguiente**	**$D_3D_2D_1D_0$**
A	0000	→ B	0001
B	0001	→ D	0011
C	0010	→ B	0001
D	0011	→ I	1000
E	0100	→ D	0011
F	**0101**	→ **F**	**0101**
G	0110	→ H	0111
H	0111	→ M	1100
I	1000	→ J	1001
J	1001	→ M	1100
K	1010	→ N	1101
L	1011	→ N	1101
M	1100	→ O	1110
N	1101	→ P	1111
O	1110	→ F	0101
P	1111	→ F	0101

Cuadro 3: Tabla de Transiciones de Estado para la Lógica del Próximo Estado

2. **Mapas de Karnaugh:**

   a) Karnaugh de D_3:

Q1Q0 / Q3Q2	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	1	0	0
10	1	1	1	1

Figura 4: K(D_3) = $Q_3\overline{Q}_2 + Q_3\overline{Q}_1 + \overline{Q}_3Q_1Q_0$

   b) Karnaugh de D_2:

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	1	1
10	0	1	1	1

Figura 5: K(D_2) = $Q_3Q_1 + Q_3Q_0 + Q_3Q_2 + Q_2Q_1 + Q_2Q_0$

c) Karnaugh de D_1:

Q3Q2 \ Q1Q0	00	01	11	10
00	0	1	0	0
01	1	0	0	1
11	1	1	0	0
10	0	0	0	0

Figura 6: K(D_1) = $Q_3Q_2\overline{Q}_1 + \overline{Q}_3Q_2\overline{Q}_0 + \overline{Q}_3\overline{Q}_2\overline{Q}_1Q_0$

d) Karnaugh de D_0:

Q3Q2 \ Q1Q0	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	1	1	1
10	1	0	1	1

Figura 7: K(D_0) = $Q_3Q_1 + \overline{Q}_2\overline{Q}_0 + \overline{Q}_3\overline{Q}_0 + \overline{Q}_3\overline{Q}_1 + Q_3Q_2Q_0$

3. **Funciones Booleanas minimizadas:**
   $f(D_3) = Q_3\overline{Q}_2 + Q_3\overline{Q}_1 + \overline{Q}_3Q_1Q_0$
   $f(D_2) = Q_3Q_1 + Q_3Q_0 + Q_3Q_2 + Q_2Q_1 + Q_2Q_0$
   $f(D_1) = Q_3Q_2\overline{Q}_1 + \overline{Q}_3Q_2\overline{Q}_0 + \overline{Q}_3\overline{Q}_2\overline{Q}_1Q_0$
   $f(D_0) = Q_3Q_1 + \overline{Q}_2\overline{Q}_0 + \overline{Q}_3\overline{Q}_0 + \overline{Q}_3\overline{Q}_1 + Q_3Q_2Q_0$

- **Decodificador de display:** Este bloque convierte el estado actual en las señales apra encender el display de 7 segmentos.

  1. **Tabla de verdad:**

Cuadro 4: Tabla de Verdad del Decodificador de Display de 7 Segmentos

Entrada (Estado Actual)					**Símbolo**	**Salida (Segmentos)**						
Nodo	**Q_3**	**Q_2**	**Q_1**	**Q_0**	||| **a**	**b**	**c**	**d**	**e**	**f**	**g**
A	0	0	0	0	A	1	1	1	0	1	1	1
B	0	0	0	1	b	0	0	1	1	1	1	1
C	0	0	1	0	C	1	0	0	1	1	1	0
D	0	0	1	1	d	0	1	1	1	1	0	1
E	0	1	0	0	E	1	0	0	1	1	1	1
F	0	1	0	1	F	1	0	0	0	1	1	1
G	0	1	1	0	G	1	0	1	1	1	1	0
H	0	1	1	1	H	0	1	1	0	1	1	1
I	1	0	0	0	I	0	1	1	0	0	0	0
J	1	0	0	1	J	0	1	1	1	1	0	0
K	1	0	1	0	K	1	0	1	0	1	1	1
L	1	0	1	1	L	0	0	0	1	1	1	0
M	1	1	0	0	M	1	0	1	0	1	0	1
N	1	1	0	1	N	1	0	1	0	1	0	0
O	1	1	1	0	O	1	1	1	1	1	1	0
P	1	1	1	1	P	1	1	0	0	1	1	1

  2. **Mapas de Karnaugh:**

Figura 8: Mapas de Karnaugh y Funciones Minimizadas para el Decodificador de Display (a a g)

Figura 9: *
$f(a) = Q_3Q_2 + Q_2\overline{Q}_1 + Q_1\overline{Q}_0 + \overline{Q}_3\overline{Q}_0$

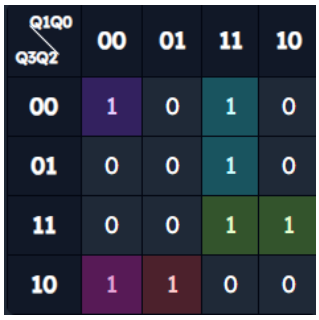


Figura 10: *
$f(b) = Q_3\overline{Q}_2\overline{Q}_1 + Q_3Q_2Q_1 + \overline{Q}_3Q_1Q_0 + \overline{Q}_2\overline{Q}_1\overline{Q}_0$

Figura 11: *
$f(c) = Q_3\overline{Q}_0 + Q_3\overline{Q}_1 + \overline{Q}_2\overline{Q}_1 + \overline{Q}_3Q_2Q_1 + \overline{Q}_3Q_1Q_0$

Figura 12: *
$f(d) = \overline{Q}_2Q_0 + Q_2Q_1\overline{Q}_0 + \overline{Q}_3Q_2\overline{Q}_0 + \overline{Q}_3Q_1\overline{Q}_0$

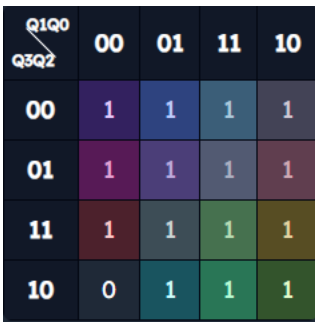


Figura 13: *
$f(e) = Q_2 + Q_1 + Q_0 + \overline{Q}_3$

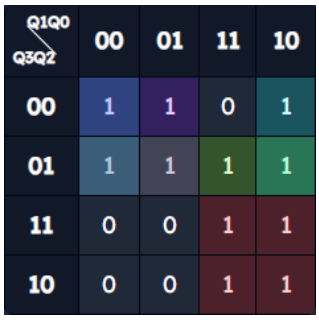


Figura 14: *
$f(f) = Q_3Q_1 + \overline{Q}_3Q_2 + \overline{Q}_3\overline{Q}_0 + \overline{Q}_3\overline{Q}_1$

Q3Q2 \ Q1Q0	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	1	0	1	0
10	0	0	0	1

Figura 15: *

$$f(g) = \overline{Q}_3 Q_0 + \overline{Q}_3 \overline{Q}_1 + Q_2 Q_1 Q_0 + Q_2 \overline{Q}_1 \overline{Q}_0 + Q_3 \overline{Q}_2 Q_1 \overline{Q}_0$$

3.2. Implementación, Arquitectura y justificación

La arquitectura del sistema de rastreo se diseño como un autómata de estados finitos del tipo Moore, implementando exclusivamente con Flip-Flops tipos D y lógica combinacional como lo exige las restricciones. El circuito principal se compone de la interconexión de tres subcircuitos modularas:

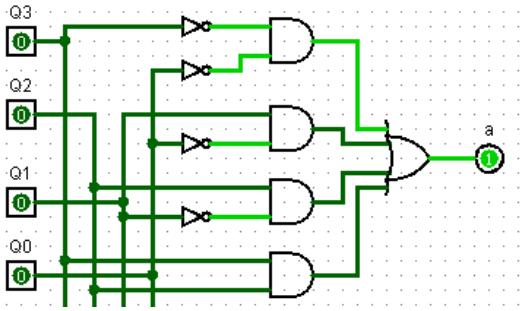
- **Registro de estado:** Encargado de almacenar el estado actual y compuesto de 4 Flip-Flops conectados en paralelo
- **Lógica de transición:** Calcula el próximo estado según la ubicación actual y el grafo de rutas. Implementado con 4 funciones booleanas obtenidas de su tabla.
- **Decodificador del display** Convierte el estado actual en la representación visual del nodo. Confeccionado con 7 funciones booleanas minimizadas a partir de su tabla de verdad.

3.2.1. Decodificador del display:

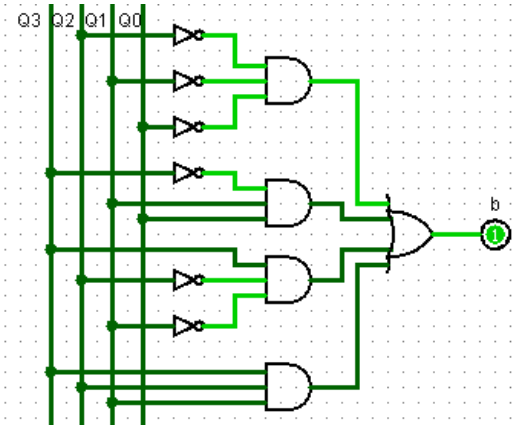
El circuito consta de 7 secciones, cada una dedicada a el segmento del display especifico y otorgando una salida correspondiente al estado de encendido del segmento, todas ellas comparten las 4 entradas booleanas donde se ingresaran los códigos de los criminales. estos fueron creados a partir de compuertas **OR, AND y NOT** por sus cualidades directas y completas para la álgebra booleana.
circuito segmentado

Para mayor comprensión del circuito, se muestra a continuación cada segmento por separado:

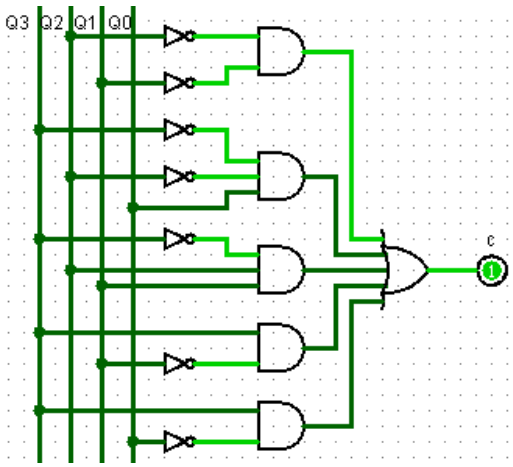
- **Segmento a:**



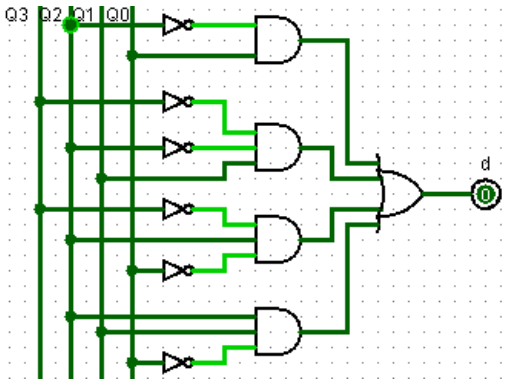
- **Segmento b:**



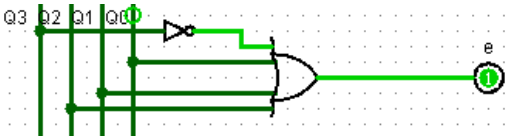
- **Segmento c:**



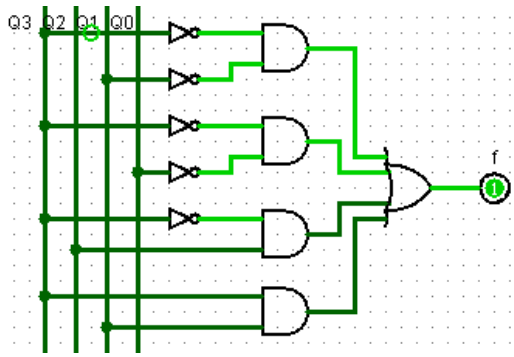
- **Segmento d:**



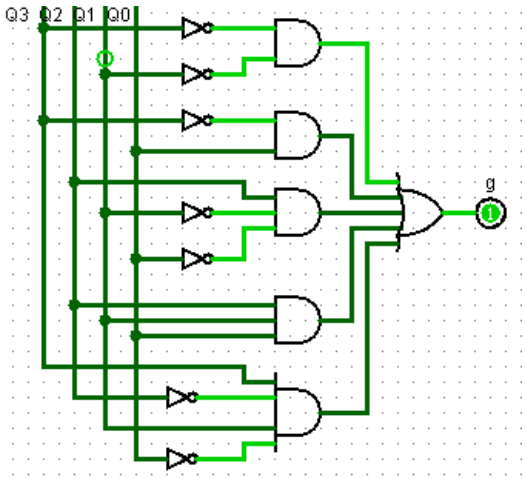
- **Segmento e:**



- **Segmento f:**



- **Segmento g:**



Arquitectura, Explicación y Fundamentos

La Arquitectura del decodificador se basa en el diseño de un **circuito combinacional**. Esto debido a que la salida, que son los símbolos entregados por el display, depende **únicamente** de la combinación actual de las 4 entradas. la utilización exclusiva de **compuertas básicas** (**AND**, **OR**, **NOT**) pareció una opción ideal para el desarrollo del sistema.

¿Por qué solo compuertas AND, OR y NOT?

Utilizamos una estrategia de **minimización óptima** para el sistema de cuatro variables que nos garantizaba obtener una expresión mínima de **Suma de Productos (SOP)**, dándonos un **álgebra booleana** sin términos

redundantes ordenados por bloques fácilmente integrables al sistema con estos tipos de compuertas. Las expresiones nos otorgaban una **guía directa y óptima** para la creación del circuito, la cual se hubiera vuelto más compleja de utilizar otras compuertas como XOR o NAND.

El **diseño del circuito** fue planeado de manera **segmentada** (modular), y cada segmento se materializó en **tres niveles de compuertas** gracias al formato **SOP**.

- **Primer nivel:** Este nivel se compone de compuertas **NOT**, **invirtiendo** todas las entradas que deban ser negadas ($\overline{A}_x$) antes de pasar al siguiente nivel.
- **Segundo nivel:** Este nivel se compone de compuertas **AND**, obtenida de las multiplicaciones booleanas que nos otorga el K-map.
- **Tercer nivel:** Por último, este y último nivel toma todos los valores obtenidos de los niveles anteriores y les aplica una compuerta **OR**. Esta es la **compuerta final** de cada segmento.

Circuito Integrado

Al completar el circuito, lo conectamos a un **display de 7 segmentos**, y se logra comprobar la **funcionalidad del decodificador** al ingresar los códigos de 4 bits y obtener la figura esperada como se muestra en las siguientes figuras.

Figura 16: Cuadrícula de 16 símbolos de 7 segmentos (4x4).

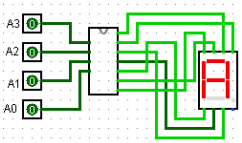


Figura 17: * 0000

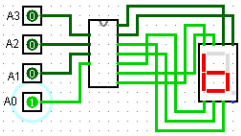


Figura 18: * 0001

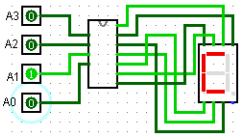


Figura 19: * 0010

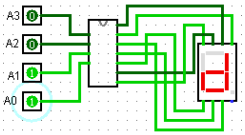


Figura 20: * 0011

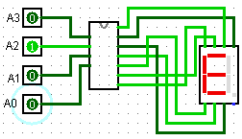


Figura 21: * 0100

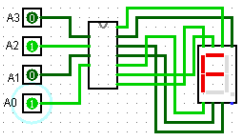


Figura 22: * 0101

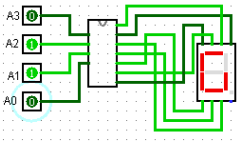


Figura 23: * 0110

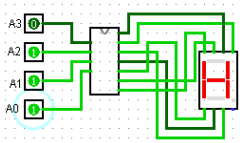


Figura 24: * 0111

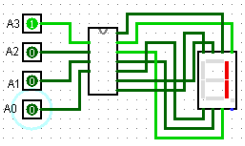


Figura 25: * 1000

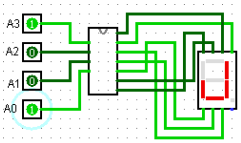


Figura 26: * 1001

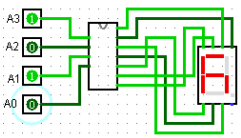


Figura 27: * 1010

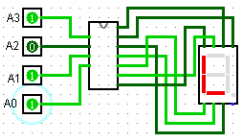


Figura 28: * 1011

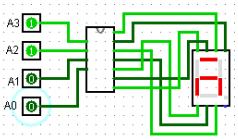


Figura 29: * 1100

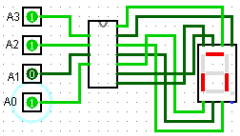


Figura 30: * 1101

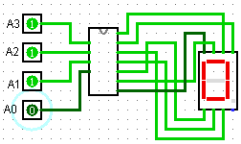


Figura 31: * 1110

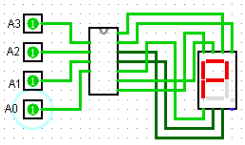


Figura 32: * 1111

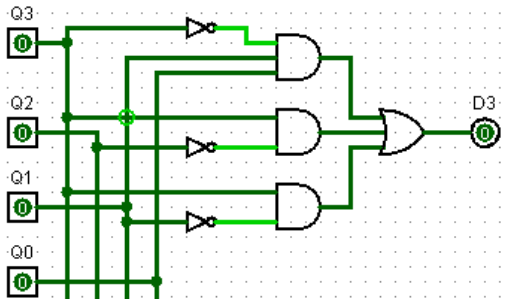
Nota: Para un efecto visual mas claro, se utilizo el circuito en formato modulo de 4 bit de entrada y 7 de salida. el circuito interior de este modulo es exactamente igual que el mostrado con anterioridad.

3.2.2. Transicionador de estados:

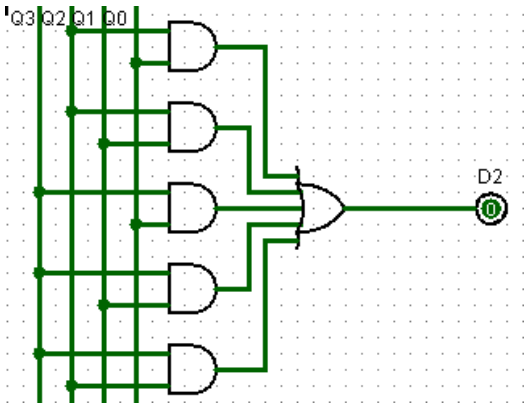
El circuito cuenta con 4 bits de entradas y 4 bits de salida, representando el estado actual y el siguiente respectivamente. Se conforma de 4 segmentos definidos para cada bit de salida como se muestra en las siguiente figuras:

Circuito segmentado: Para mayor comprensión se muestra los segmentos por separado.

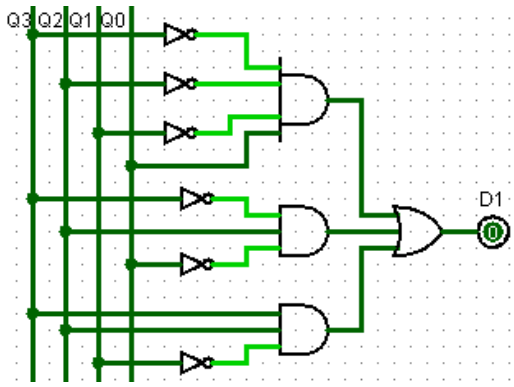
- **Segmento D_3:**



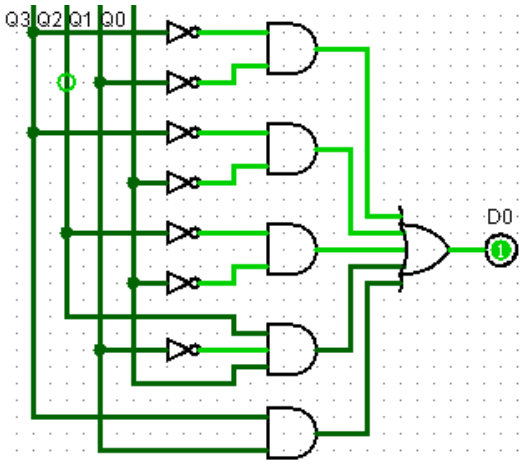
- **Segmento D_2:**



- **Segmento D_1:**



- **Segmento D_0:**



Arquitectura, Explicación y Fundamentos

Este subcircuito constituye el corazón combinacional del Autómata de Estados Finitos (FSM). Su función es calcular el **Próximo Estado** ($\mathbf{D_3D_2D_1D_0}$) basándose únicamente en el **Estado Actual** ($\mathbf{Q_3Q_2Q_1Q_0}$), siguiendo las transiciones definidas por el Diagrama de Estados (Figura 3).

1. **Implementación:** Es un circuito puramente combinacional, implementado mediante una estructura de **Suma de Productos (SOP)** de dos niveles (AND-OR), utilizando las funciones booleanas minimizadas obtenidas de la Tabla de Transiciones (Cuadro 3).

2. **Lógica de Detención:** El diseño combina el flujo de rastreo con la condición de detención. Al ser implementado con las funciones minimizadas, el subcircuito garantiza que si la entrada es el nodo de extracción **F(0101)**, la salida $\mathbf{D_{next}}$ también será **0101**, logrando que el sistema se detenga automáticamente al alcanzar dicho estado.

3.2.3. Control de carga

Es un circuito simple que permite controlar que se esta cargando al sistema, si es el valor inicial o si es algún valor intermedio, consta de 8 bits de entrada ($A_3A_2A_1A_0$ y $D_3D_2D_1D_0$) Ademas de un bit de control que nos permitirá reiniciar cuando estimemos conveniente, y 4 bits de salida ($D'_3D'_2D'_1D'_0$) indicando la ubicación que se esta manejando actualmente. Este subcircuito posee el siguiente diseño:

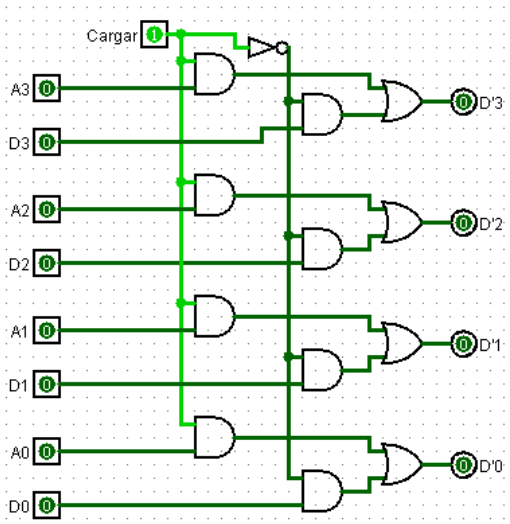


Figura 33: subcircuito de control de estado

Arquitectura, Explicación y Fundamentos

Este subcircuito es el encargado de gestionar el flujo de datos que ingresa al **Registro de Estado** (Flip-Flops D), permitiendo alternar entre el modo de **Inicialización** y el modo de **Rastreo Secuencial**.

1. **Función y Arquitectura:** El bloque se implementa como un **Multiplexor (MUX) de 4 bits** de 2 a 1, construido exclusivamente con

lógica combinacional (AND, OR, NOT). Su propósito es seleccionar qué valor se carga en la entrada D de los Flip-Flops (D').

2. **Mecanismo de Control:** La entrada de control `Cargar` determina el modo de operación:
   - Si `Cargar` $= \mathbf{1}$, el MUX selecciona el **Código de Entrada** ($\mathbf{A_3A_2A_1A_0}$), cargando el nodo inicial.
   - Si `Cargar` $= \mathbf{0}$, el MUX selecciona el **Próximo Estado** ($\mathbf{D_{next}}$) calculado por la Lógica de Transición, permitiendo el avance secuencial.
3. **Fórmula Implementada:** La lógica interior obedece a la ecuación del MUX, aplicada bit a bit (i):

$$D'_i = (\overline{\texttt{Cargar}} \cdot D_{next,i}) + (\texttt{Cargar} \cdot A_i)$$

3.2.4. Circuito final (Main)

Ahora con los elementos que ya tenemos, somos capaces de elaborar el circuito completo que se encontrara en el main. este constara de las 4 entradas de la ubicación inicial ($A_3A_2A_1A_0$), un bit de control para los estados junto a el **CLK** que se encargara de las actualizaciones. y como salida tendremos la representación de la ubicación mediante un display de 7 segmentos. El circuito queda ilustrado a continuación

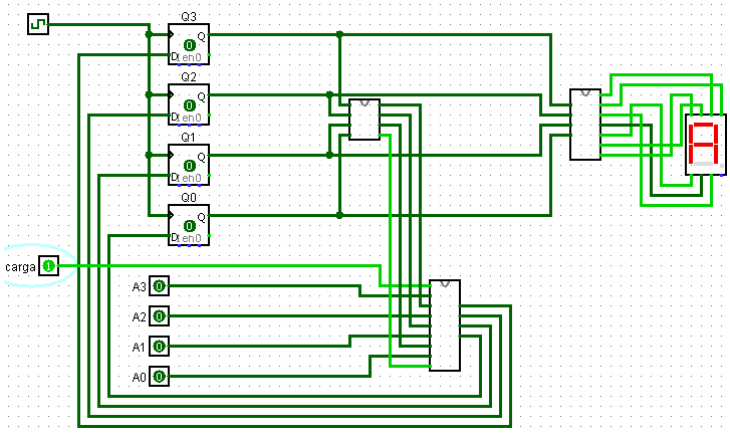


Figura 34: Circuito completo del Rastreador de criminales

Arquitectura, Explicación y Fundamentos
El circuito principal (`main`) no es simplemente una colección de componentes, sino una **arquitectura secuencial modular** diseñada para optimizar la claridad, depuración y cumplimiento de los requisitos del sistema de rastreo. Cada decisión en su ensamblaje se justifica por principios de diseño de autómatas de estados finitos y la eficiencia en la implementación.

1. **Entradas del Sistema y Control Centralizado:**
    - **Código de Entrada ($\mathbf{A_3A_2A_1A_0}$):** Constituye la entrada inicial del usuario, estableciendo el punto de partida. Se ubica en el `main` como un bus de 4 bits para facilitar la configuración inicial.
    - **Señal de Control (`Cargar`):** Este bit es fundamental para la **flexibilidad operativa**. Su inclusión permite alternar explícitamente entre la carga de un nuevo estado inicial y el avance automático por la ruta. Sin este bit, el sistema requeriría una lógica de detección de estado especial o un mecanismo de reinicio más complejo, lo que añadiría complejidad innecesaria a la Lógica de Transición o a los Flip-Flops.
    - **Reloj (`CLK`):** Como todo sistema secuencial síncrono, un único `CLK` global es imprescindible. Se centraliza en el `main` para asegurar que todas las transiciones de estado en los Flip-Flops ocurran de manera coordinada.
2. **Registro de Estado: El Corazón Secuencial Síncrono:**
    - Cuatro **Flip-Flops tipo D** forman el registro de estado, almacenando $\mathbf{Q_3Q_2Q_1Q_0}$. La elección de FF-D es directa, ya que su entrada D es simplemente el próximo estado deseado, simplificando la lógica combinacional previa.
3. **Interconexión Modular y Flujo de Datos:**
    - **Bucle de Realimentación:** Las salidas $\mathbf{Q_3Q_2Q_1Q_0}$ de los Flip-Flops se realimentan al subcircuito Transicionador de estados y Decodificador de Display. Esta realimentación es la esencia de un autómata de estados finitos, donde el estado futuro depende del estado presente, y la visualización refleja precisamente ese estado.

4. **Visualización Clara y Modularidad de Salida:**

    - El subcircuito `Decodificador_Display` toma el estado actual ($\mathbf{Q_{actual}}$) y genera las señales para un **Display de 7 segmentos**. Esta modularidad es crucial; el circuito principal no necesita conocer los detalles de cómo se codifica cada símbolo en el display, solo que el módulo lo hará. Esto permite futuras modificaciones en la representación visual sin alterar la lógica de estado.

En síntesis, el ensamblaje del `main` es un reflejo de un diseño jerárquico y modular. Cada subcircuito cumple una función específica, lo que mejora la legibilidad, facilita la depuración y demuestra una clara separación de las preocupaciones de diseño (control, lógica de estado, almacenamiento y visualización).

4. Validaciones

Por ultimo, se registrara a continuación varias pruebas en distintas ubicaciones iniciales para mostrar el correcto funcionamiento del circuito:

- **Recorrido iniciando en A:**

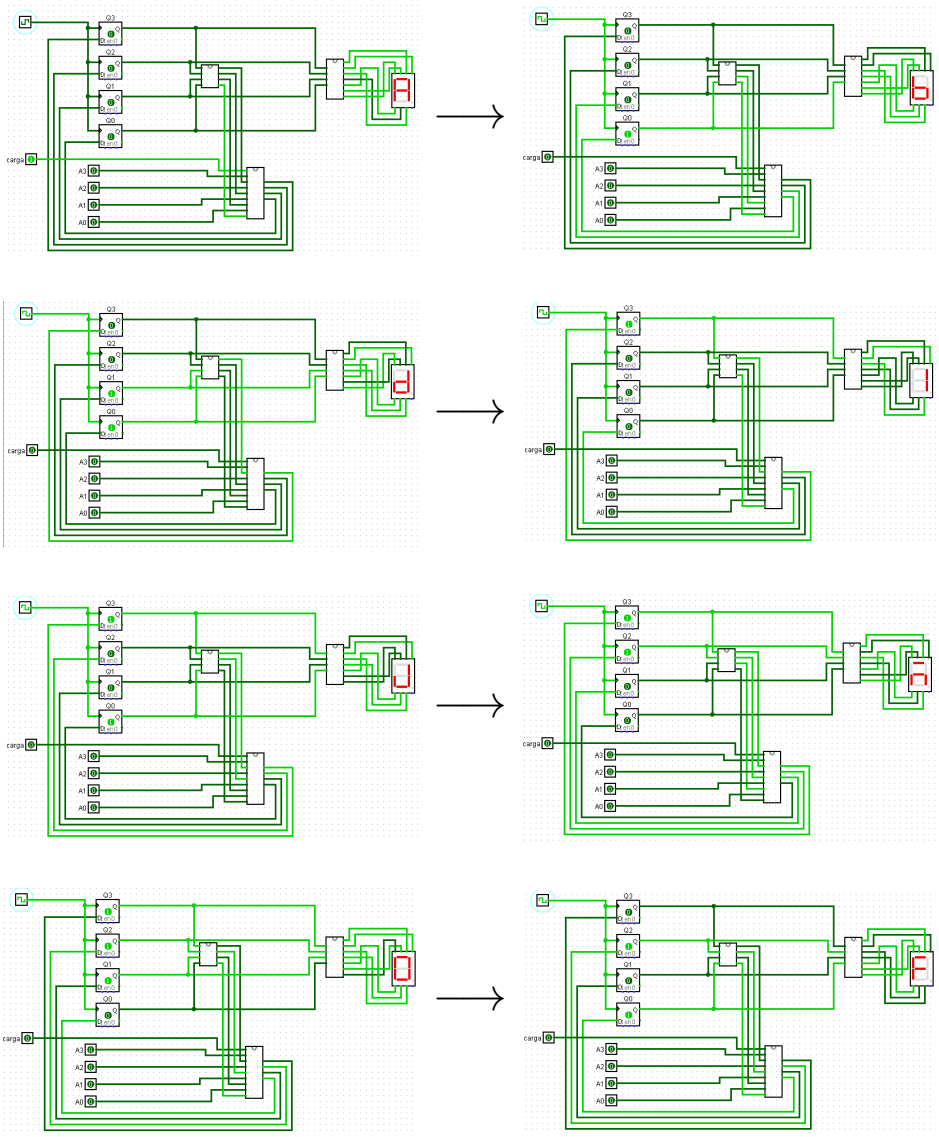


Figura 35: Recorrido del rastreador iniciando en A hasta F.

- **Recorrido iniciando en C:**

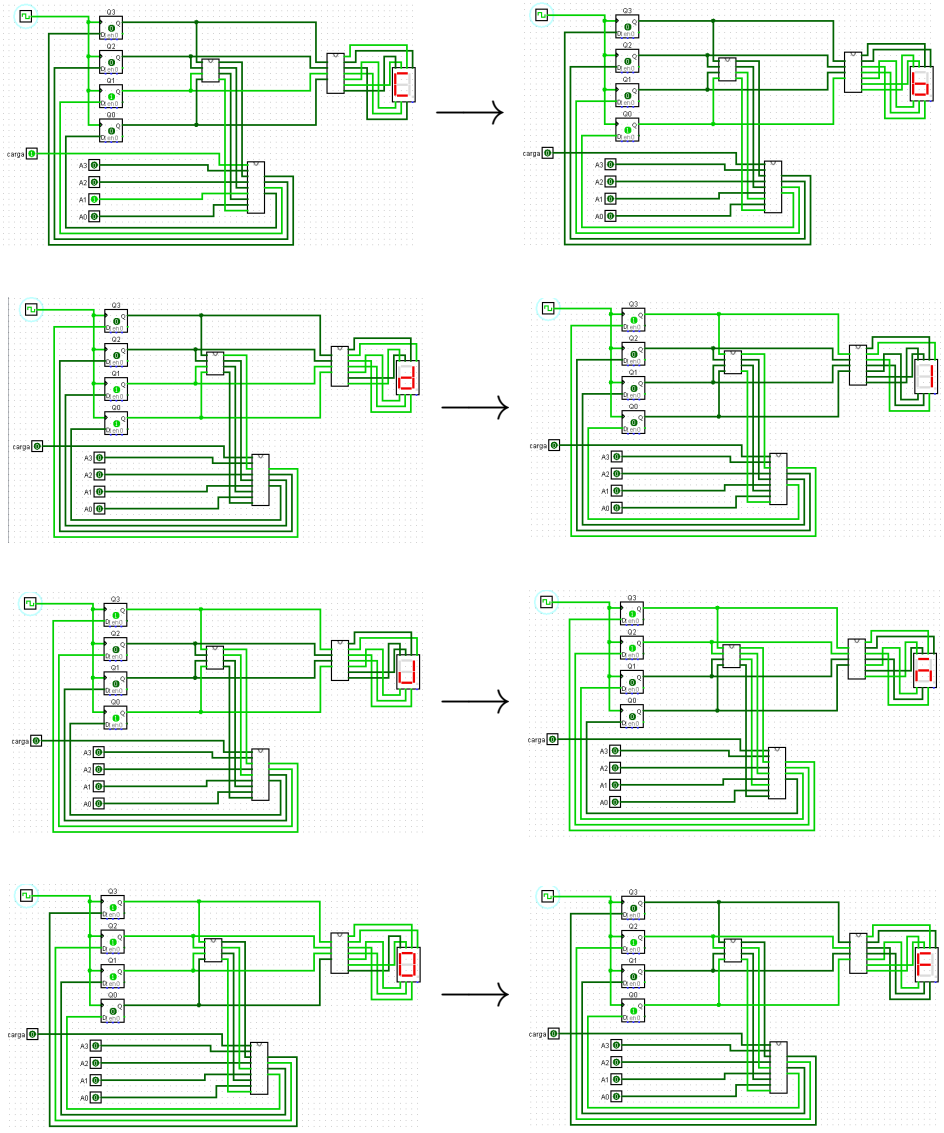


Figura 36: Recorrido del rastreador iniciando en C hasta F.

- **Recorrido iniciando en L:**

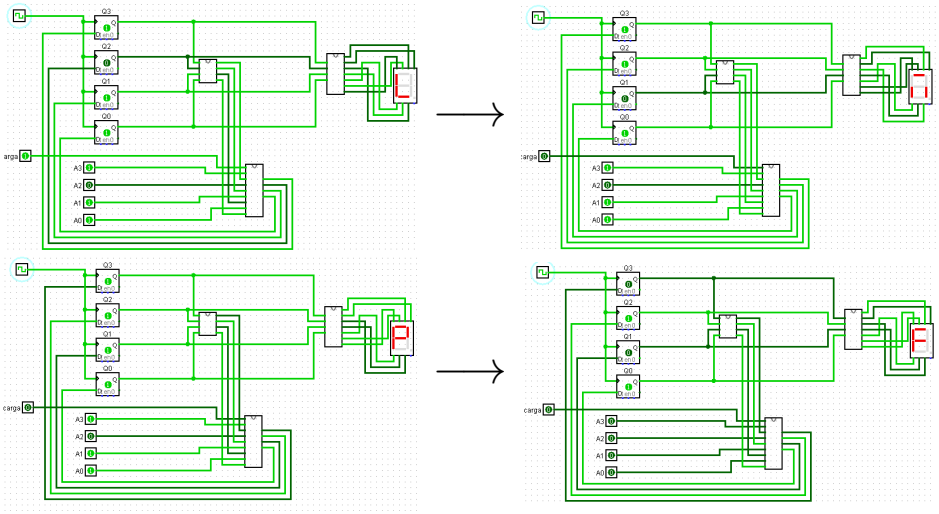


Figura 37: Recorrido del rastreador iniciando en L hasta F.

- **Recorrido iniciando en G:**

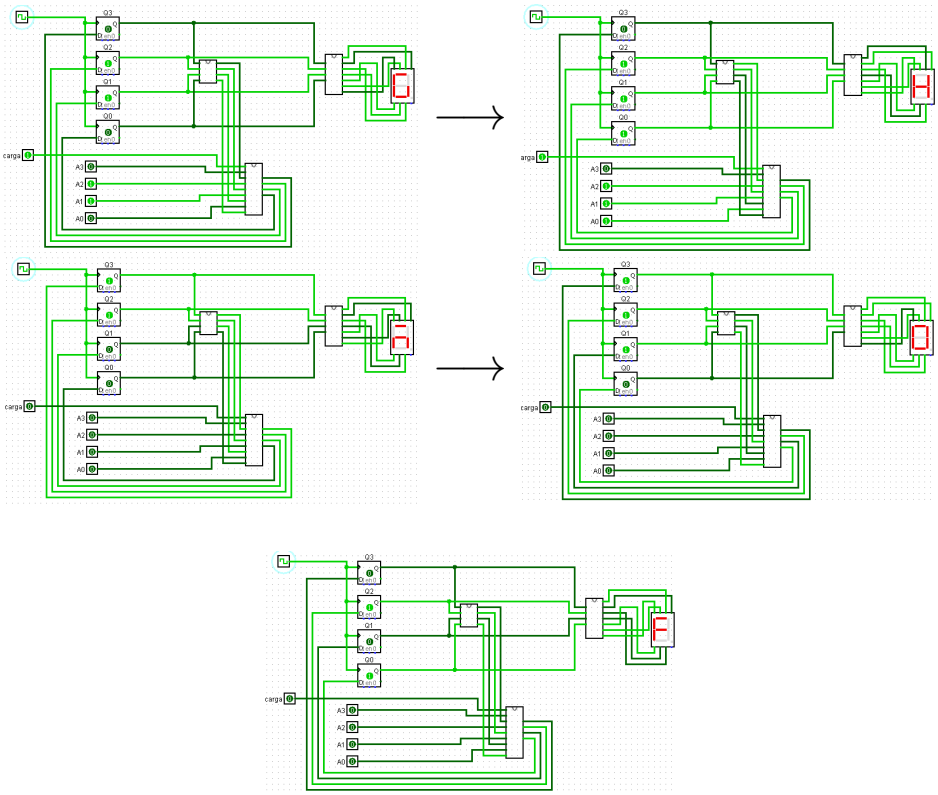


Figura 38: Recorrido del rastreador iniciando en G hasta F.

4.1. Recorridos Esperados del Rastreador

Por ultimo, la siguiente tabla resume los recorridos que el sistema de rastreo de criminales sigue para cada ubicación inicial dada.

Ubicación Inicial	**Recorrido Esperado hasta F**
A (0000)	A $\longrightarrow$ B $\longrightarrow$ D $\longrightarrow$ I $\longrightarrow$ J $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
B (0001)	B $\longrightarrow$ D $\longrightarrow$ I $\longrightarrow$ J $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
C (0010)	C $\longrightarrow$ B $\longrightarrow$ D $\longrightarrow$ I $\longrightarrow$ J $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
D (0011)	D $\longrightarrow$ I $\longrightarrow$ J $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
E (0100)	E $\longrightarrow$ D $\longrightarrow$ I $\longrightarrow$ J $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
F (0101)	F (Detención inmediata)
G (0110)	G $\longrightarrow$ H $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
H (0111)	H $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
I (1000)	I $\longrightarrow$ J $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
J (1001)	J $\longrightarrow$ M $\longrightarrow$ O $\longrightarrow$ F
K (1010)	K $\longrightarrow$ N $\longrightarrow$ P $\longrightarrow$ F
L (1011)	L $\longrightarrow$ N $\longrightarrow$ P $\longrightarrow$ F
M (1011)	M $\longrightarrow$ O $\longrightarrow$ F
N (1011)	N $\longrightarrow$ P $\longrightarrow$ F
O (1011)	O $\longrightarrow$ F
P (1011)	P $\longrightarrow$ F

Cuadro 5: Recorridos del Rastreador desde Ubicaciones Iniciales hasta F