Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

# CIRCUITOS DIGITALES AVANZADOS

## Practica 4
## Memorias

**Docente:** Lara Camacho Evangelina

**Alumnos:**
Gómez Cárdenas Emmanuel Alberto **1261509**
León Romero Pablo Constantino **1253171**

# INDICE

# OBJETIVO
Diseñar y construir un circuito detector de secuencia usando una memoria para la función de transición y salida de la máquina de estados.

# EQUIPO
Computadora personal con el software Logisim.

# FUNDAMENTO TEORICO

## Memorias no volátiles
Las memorias de solo lectura (Read-only memory, ROM) son un tipo de memoria semiconductora diseñadas para almacenar datos que son permanentes o que no van a cambiar frecuentemente. Durante la operación normal, no se escriben datos nuevos en la ROM, pero los datos que ya están almacenados pueden leerse de ella. En algunas ROMs los datos son ingresados cuando son manufacturadas; en otras los datos se pueden ingresar eléctricamente. El proceso de ingresar datos es llamado programar o quemar la ROM. En algunas ROMs no se pueden modificar los datos que ya fueron programados, otras pueden ser borradas y reprogramadas varias veces.

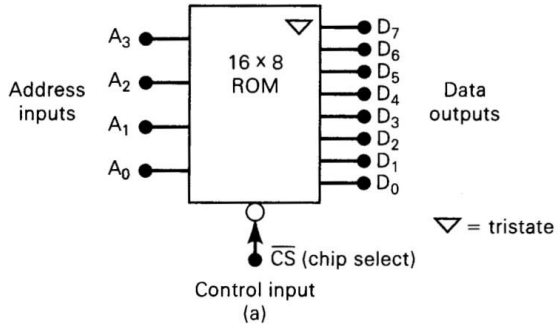
Las ROM son usadas para almacenar datos que no van a cambiar durante la operación normal del sistema. Un uso común de las ROM es para almacenar programas en las computadoras. Debido a que son no volátiles, los programas no se pierden cuando se quita la alimentación de energía a la computadora.

## Arquitectura de una ROM
La Fig. 1 muestra el diagrama a bloques de una ROM. Se pueden distinguir tres conjuntos de señales: líneas de dirección, líneas de control y líneas de datos. Las líneas de dirección y de control son entradas y las líneas de datos son salidas en una ROM. En memorias que no son de solo lectura, las líneas de datos son bidireccionales.

- Líneas de dirección: Seleccionan la localidad o registro de la memoria a leer. La cantidad de localidades en una memoria está dada por la relación 2N, donde N es el número de líneas de dirección en el dispositivo. En la Fig. 1, las líneas de dirección son 4: A0 – A3, por lo tanto, la memoria tiene 24 = 16 localidades.

- Líneas de datos: Proveen los datos almacenados en una localidad de memoria. En la Fig. 1, las líneas de datos son D0 – D7, por lo tanto, cada localidad almacena 8 bits. La capacidad de la memoria es de 16 x 8 bits.

- Líneas de control: Permiten habilitar la memoria, habilitar las salidas, seleccionar el modo de lectura (y/o de escritura si la memoria no es de solo lectura). En la Fig. 1, la línea de control es CS (Chip Select), la cual se activa en bajo. La memoria solo provee las salidas cuando está línea está en bajo. Cuando se encuentra en alto, las líneas de datos están en alta impedancia, Hi-Z.




| | Address | | | | Data | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | $A_3$ | $A_2$ | $A_1$ | $A_0$ | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 13 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |

(b)

| | Address | Data |
|---|---|---|
| Word | $A_3$ $A_2$ $A_1$ $A_0$ | $D_7$–$D_0$ |
| 0 | 0 | DE |
| 1 | 1 | 3A |
| 2 | 2 | 85 |
| 3 | 3 | AF |
| 4 | 4 | 19 |
| 5 | 5 | 7B |
| 6 | 6 | 00 |
| 7 | 7 | ED |
| 8 | 8 | 3C |
| 9 | 9 | FF |
| 10 | A | B8 |
| 11 | B | C7 |
| 12 | C | 27 |
| 13 | D | 6A |
| 14 | E | D2 |
| 15 | F | 5B |

(c)

Figura 1. a) Diagrama a bloques de una ROM; b) tabla mostrando los datos en cada dirección de la ROM en formato binario; c) similar a la tabla anterior pero los datos en formato hexadecimal.

### Procedimiento de lectura:

Colocar la dirección a leer en las líneas de dirección A0 – A3.
Colocar la señal de habilitación del dispositivo, línea CS en bajo.
La memoria provee en las líneas de datos D0 – D7 el dato que está almacenado en la localidad seleccionada en el paso 1.

### Tiempo de respuesta:

Al hacer una lectura de una ROM, existe un retardo de propagación de señales entre la aplicación de las entradas a la ROM para la lectura y la salida de los datos. Este retardo, llamado tiempo de acceso, tACC, es una medición de la velocidad de operación de la

ROM. La Fig. 2 muestra el tiempo de acceso. Justo antes de t1 las direcciones están cambiando para una lectura en una nueva dirección. En t1 la dirección es válida, esto es, cada línea de dirección está en un nivel lógico válido. En este punto, la ROM empieza a decodificar la dirección para seleccionar la localidad que va a enviar sus datos a los buffers de salida. En t2, CS es activado para habilitar los buffers de salida. Finalmente, en t3, las salidas cambian de un estado de alta impedancia a los niveles que representan los datos almacenados en la localidad específica.

El tiempo entre t1 y t3 es el tiempo de acceso tACC. ROMs bipolares típicas tienen un tACC entre 30 a 90 ns, dispositivos NMOS de 35 a 500 ns, tecnologías CMOS tienen un rango de 20 a 60 ns.

Otro parámetro de tiempo importante es el tiempo en la habilitación de la salida, tOE, el cual es el retardo entre la activación de CS y la salida de datos válidos. Valores típicos para tecnologías bipolares son de 10 a 20 ns, 25 a 100 ns para NMOS y 12 a 50 ns para CMOS.

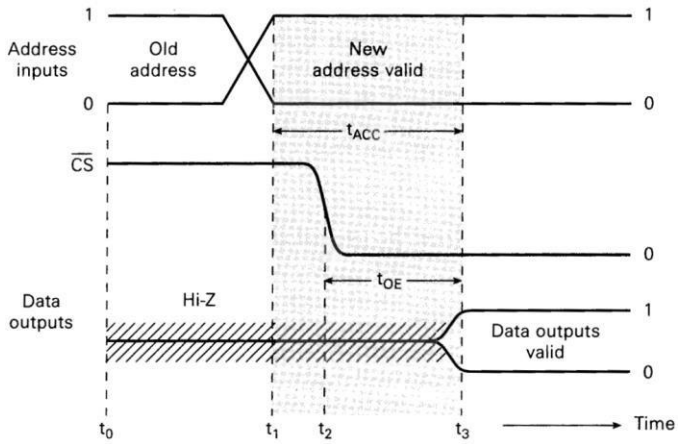
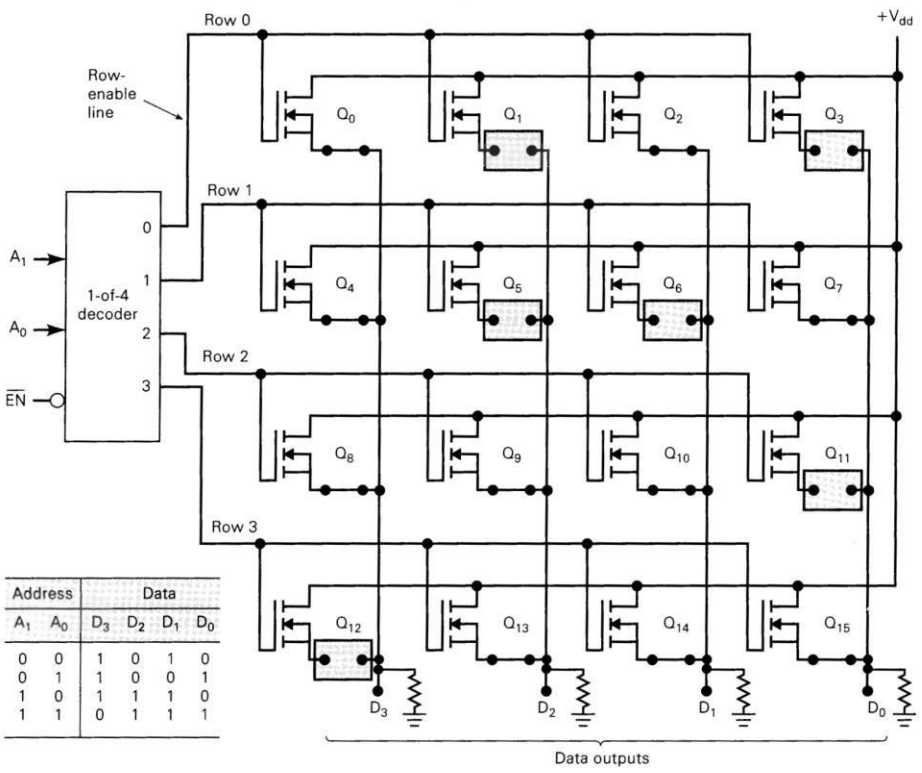


Figura 2. Tiempos típicos de una ROM en una lectura.

## Tipos de ROM:

Máscara ROM: Las localidades de memoria son escritas en tiempo de manufactura.
Una máscara especial es usada para controlar las interconexiones eléctricas en el chip. Sus datos no pueden ser reprogramados. La Fig. 3 muestra la estructura de una memoria Máscara ROM.




| Address | | Data | | | |
|---|---|---|---|---|---|
| $A_1$ | $A_0$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |

Figura 3. Estructura de una Máscara ROM. Un MOSFET es usado por cada celda de memoria. Una conexión abierta almacena un 0, una conexión cerrada almacena un 1.

PROM (Programmable ROM, ROM Programable): No son programadas durante la manufactura. El usuario puede programarlas una sola vez, ya programadas no pueden ser borradas.

EPROM (Erasable Programmable ROM, ROM Borrable y Programable): Puede ser programada por el usuario y también puede ser borrada y reprogramada cuantas veces sea requerido. El proceso de programado involucra aplicar niveles especiales de voltaje (típicamente en el rango de 10 a 25 V), a algunos pines del chip por cierto tiempo (típicamente 50 ms por localidad de memoria). El programado usualmente se realiza con un circuito de programación especial separado del sistema en el cual la EPROM va a operar.

En estado normal, todos los transistores en las celdas de la EPROM están apagados y almacenan un 1 lógico. Los transistores pueden ser encendidos al aplicar un alto voltaje de programación para que la celda almacene un 0 lógico. Durante el proceso de programado, las direcciones de la EPROM y las líneas de datos son usadas para seleccionar que celdas van a ser programadas con ceros y cuáles van a permanecer con unos.

Una vez que una EPROM ha sido programada, puede ser borrada al exponerla a luz ultravioleta (UV) aplicada a través de una ventana en el empaquetado del chip, por entre 15 a 20 minutos. La luz UV apaga todos los transistores, de forma que todas las celdas

almacenan 1. Una vez borrada la memoria, ésta puede ser reprogramada. La Fig. 5 muestra un ejemplo de una EPROM y la Fig. 6 de un borrador para EPROMs por luz UV.

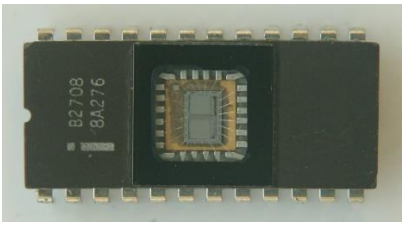



Figura 5. EPROM.


Figura 6. Luz UV para borrado de EPROMs.

EEPROM (Electrically EPROM, ROM Eléctricamente Borrable y Programable): Es una mejora de las EPROM ya que permite borrar el contenido de la memoria eléctricamente en lugar de con luz UV. La operación de borrar y almacenar nuevos datos usualmente toma 5 ms. Debido a la complejidad de sus celdas y circuitería de soporte, las memorias EEPROM suelen ser de menor capacidad y mayor costo que las EPROM.

Memoria Flash: El reto de los ingenieros de semiconductores era fabricar una memoria con las características de borrado eléctrico de las EEPROM, pero con las capacidades de memoria y costos de las EPROM, manteniendo la alta velocidad de lectura de ambas. Esto se logró con las memorias flash. El nombre de las memorias hace referencia a la rapidez con que son borradas y escritas. Muchas poseen un modo de borrado masivo donde toda la memoria es borrada, otras ofrecen un modo de borrado por sector donde solo sectores específicos de la memoria son borrados. Esto evita tener que borrar toda la memoria si solo una porción va a ser actualizada.

La Fig. 6 muestra una comparación de las memorias no volátiles respecto a cómo la flexibilidad de borrado y programado incrementan la complejidad y costo del dispositivo.

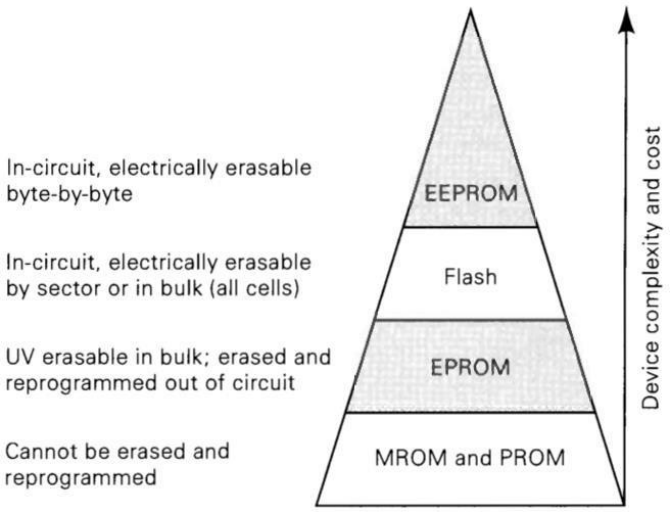



Figura 6. Comparativa de distintas memorias no volátiles respecto a su complejidad y costo.

# DESARROLLO

## Parte 1

1. Diseñe y simule un circuito detector de secuencia con una entrada **X** y una salida **Z**, que detecte la aparición de la secuencia 11001 en la entrada. La salida **Z** es 1 cada vez que la secuencia es recibida. El detector debe ser modelo Moore y con traslape. Haga uso de una memoria ROM para la función de transición y la salida de la máquina de estados. Utilice flip flops D como elemento de memoria.

### Diagrama de estados (Entrada/Salida)

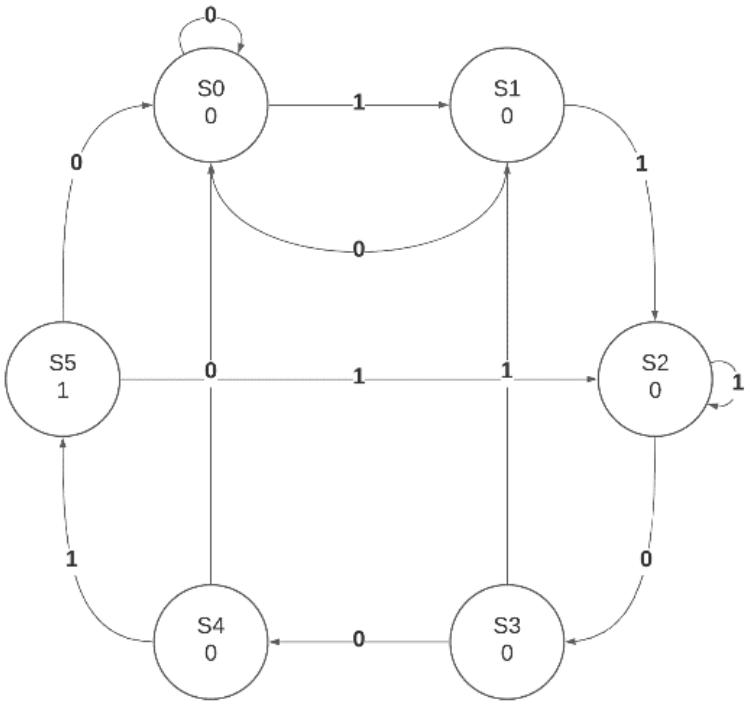



Tabla 1. Asignación de estados.

| Estado | q2q1q0 |
|---|---|
| S0 | 000 |
| S1 | 001 |
| S2 | 010 |
| S3 | 011 |
| S4 | 100 |
| S5 | 101 |
| S6 | 110 |
| S7 | 111 |

Tabla de Transición de Estados

| Tabla De transicion | | | |
|---|---|---|---|
| ESTADO | Proximo Estado (Q2, Q1, Q0) | | Salidas |
| Actual | X = 0 | X = 1 | Z0 |
| 000 | 000 | 001 | 0 |
| 001 | 000 | 010 | 0 |
| 010 | 011 | 010 | 0 |
| 011 | 100 | 001 | 0 |
| 100 | 000 | 101 | 0 |
| 101 | 010 | 000 | 1 |
| 110 | 000 | 000 | 0 |
| 111 | 000 | 000 | 0 |

Gómez Cárdenas Emmanuel Alberto
León Romero Pablo Constantino

## Tablas de Transición de estados por flip flop.

*Flip-Flop D2*

| Transicion de estados | | |
|---|---|---|
| Y2 | X = 0 | X = 1 |
| Y2Y1Y0 | Y2 | Y2 |
| 000 | 0 | 0 |
| 001 | 0 | 0 |
| 010 | 0 | 0 |
| 011 | 1 | 0 |
| 100 | 0 | 0 |
| 101 | 0 | 1 |
| 110 | 0 | 0 |
| 111 | 0 | 0 |

*Flip-Flop D1*

| Transicion de estados | | |
|---|---|---|
| Y1 | X = 0 | X = 1 |
| Y2Y1Y0 | Y1 | Y1 |
| 000 | 0 | 0 |
| 001 | 0 | 1 |
| 010 | 1 | 1 |
| 011 | 0 | 0 |
| 100 | 0 | 0 |
| 101 | 1 | 0 |
| 110 | 0 | 0 |
| 111 | 0 | 0 |

*Flip-Flop D0*

| Transicion de estados | | |
|---|---|---|
| Y0 | X = 0 | X = 1 |
| Y2Y1Y0 | Y0 | Y0 |
| 000 | 0 | 1 |
| 001 | 0 | 0 |
| 010 | 1 | 0 |
| 011 | 0 | 1 |
| 100 | 0 | 1 |
| 101 | 0 | 0 |
| 110 | 0 | 0 |
| 111 | 0 | 0 |

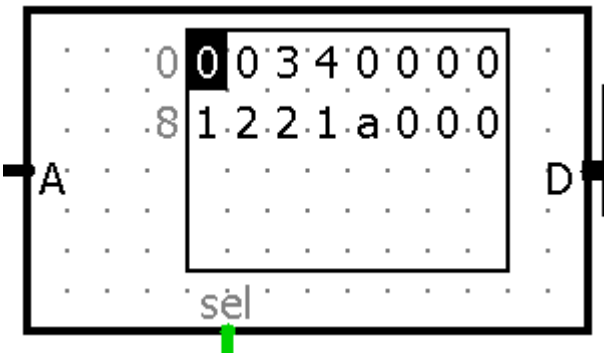
*Salida*

| Salida | |
|---|---|
| Actual | Salida |
| Q2 Q1 Q0 | Z0 |
| 000 | 0 |
| 001 | 0 |
| 010 | 0 |
| 011 | 0 |
| 100 | 0 |
| 101 | 1 |
| 110 | 0 |
| 111 | 0 |

Gómez Cárdenas Emmanuel Alberto
León Romero Pablo Constantino

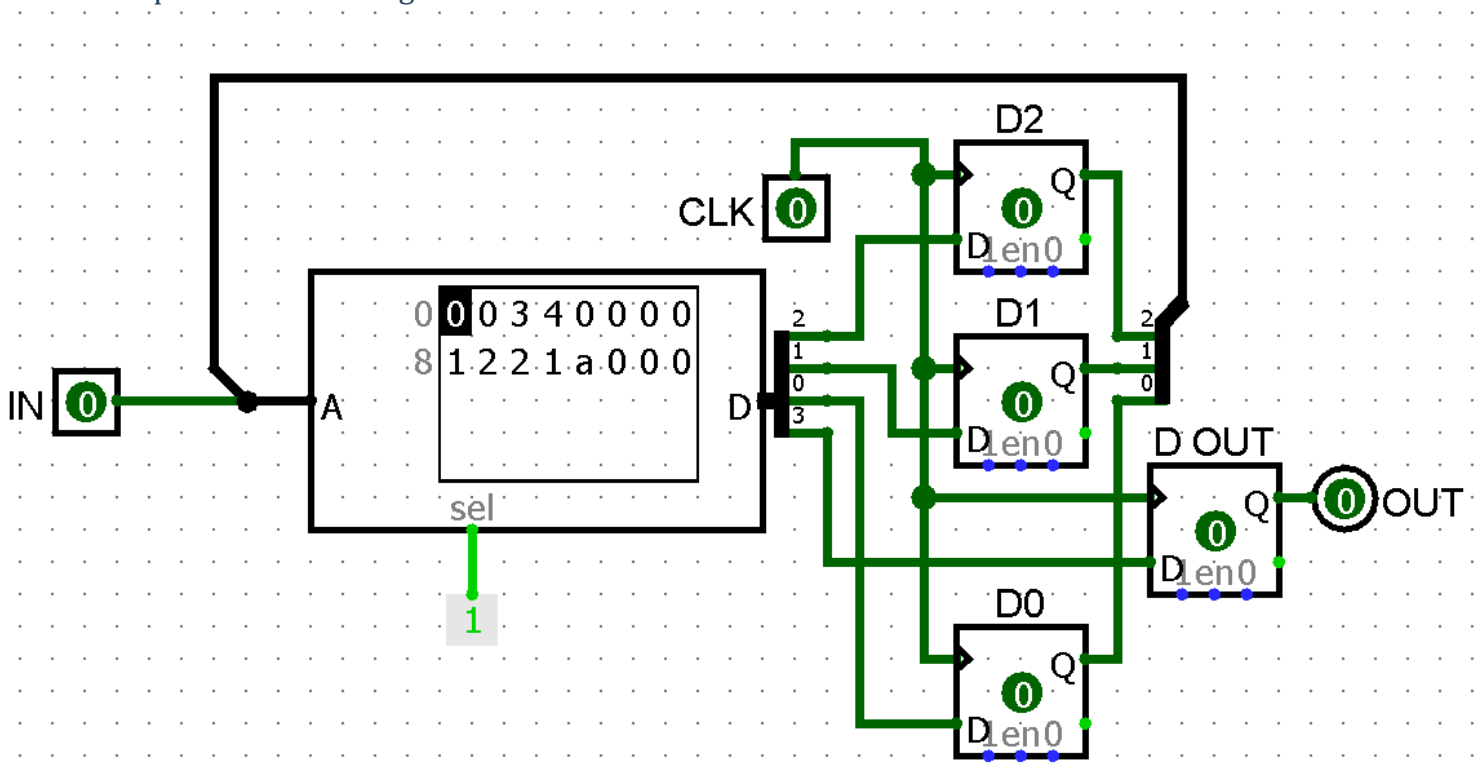
## Llenado de la ROM

La ROM utilizada es una de 4 bits en el bus de datos y 4 en el bus de direcciones, de los cuales, 3 bits son los flip-flops y el 4to es el pin de entrada y salida.
El estado actual es la dirección actual y el estado siguiente es el valor almacenado en dicha dirección




Como se pude observar contamos con dos renglones de direcciones, en todas las direcciones de arriba (0-7) se dan cuando el pin de entrada (4to bit) es 0, así mismo, cuando el bit de entada es 1 se utilizan las direcciones (8-15).
Los valores dentro de cada dirección indican el estado siguiente y el bit más significativo indica el estado del pin de salida, esto quiere decir que, si tenemos un valor mayor a 7 dentro de una dirección, el pin de salida será alimentado.

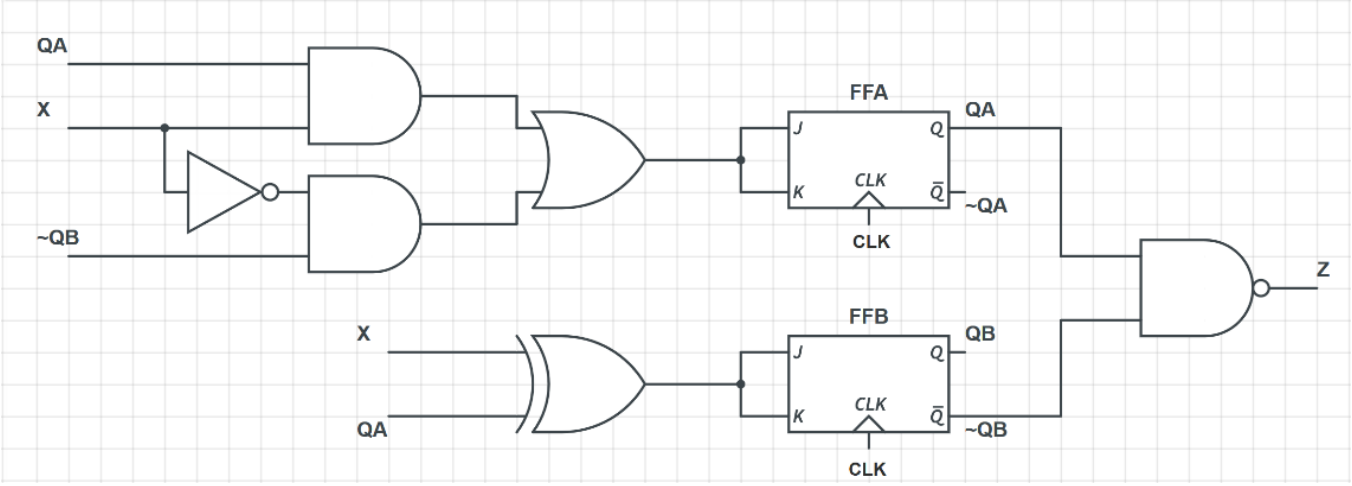
## Circuito implementado en Logisim




Se le ha agregado un flip flop a la salida para que esta dure exactamente un ciclo de reloj

## Parte 2

2. Analice el siguiente circuito y derive:
   - Ecuaciones lógicas para JA y KA (flip flop FFA), JB y KB (flip flop FFB) y Z.
   - Tabla de excitación y tabla de transición de estados.
   - Diagrama de estados.




## Ecuaciones Lógicas

Ecuacion FFA

(QA * X) + (QB' * X')

| Transicion de estados | | |
|---|---|---|
| FFA | X = 0 | X = 1 |
| QAQB | QA | QA |
| 00 | 1 | 0 |
| 01 | 0 | 0 |
| 10 | 0 | 0 |
| 11 | 1 | 0 |

Ecuacion FFB

QA XOR X

| Transicion de estados | | |
|---|---|---|
| FFB | X = 0 | X = 1 |
| QAQB | QB | QB |
| 00 | 0 | 1 |
| 01 | 1 | 0 |
| 10 | 1 | 0 |
| 11 | 0 | 1 |

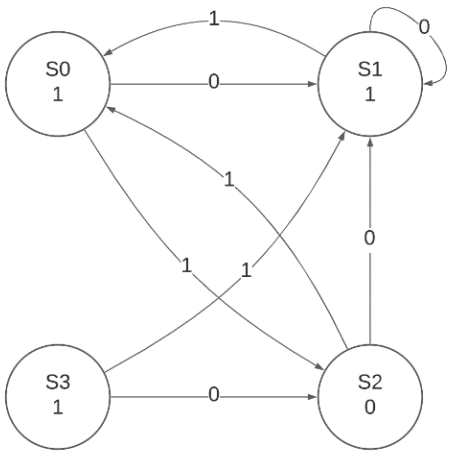
Ecuacion Z

(QA * QB')'

| Salida | |
|---|---|
| Actual | Salida |
| QAQB | Z |
| 00 | 1 |
| 01 | 1 |
| 10 | 0 |
| 11 | 1 |

## Diagrama y Transición de Estados

| Tabla De transicion | | | |
|---|---|---|---|
| ESTADO | Proximo Estado (Q2, Q1, Q0) | | Salidas |
| Actual | X = 0 | X = 1 | Z |
| 00 | 10 | 01 | 1 |
| 01 | 01 | 00 | 1 |
| 10 | 01 | 00 | 0 |
| 11 | 10 | 01 | 1 |



# CONCLUSIONES

**Gómez Cárdenas Emmanuel Alberto:**
En esta pudimos observar como una ROM nos puede facilitar bastante los cálculos y recursos a utilizar para hacer un circuito, así mismo, que tan fácil es implementarla en circuitos anteriormente creados. También nos dimos cuenta de lo interesante que es obtener un diagrama de estados desde un circuito, utilizando exactamente el mismo proceso, solamente con los procedimientos invertidos.

**Pablo Constantino León romero:**
En esta práctica se aprendió a usar una memoria ROM para remplazar a las compuertas utilizadas en prácticas pasadas. Esto nos ayuda en realizar circuitos más compactos y de manera más rápida. Además, se realizó el procedimiento, pero de manera inversa al pasar de un circuito ya realizado a su diagrama de estados, esto nos ayuda en tener un mejor entendimiento y así poder ir de un lado a otro.

# Video de Practica

https://drive.google.com/file/d/1YQ154VYGNufkrMA7h_Fz_W1-B1xl-Loc/view