# Informe

**Laboratorio 1**

Luis Ángel Vanegas Martinez

Problema #2

Diseño de Circuitos Combinacionales

Arquitectura de Computadores y Laboratorio

Fredy Alexánder Rivera Vélez

Universidad de Antioquia

2021-1

# Objetivo

* Emplear los conocimientos teóricos adquiridos en el curso en el proceso de diseño de sistemas digitales combinacionales.
* Emplear herramientas de software para el diseño y la simulación de sistemas digitales.

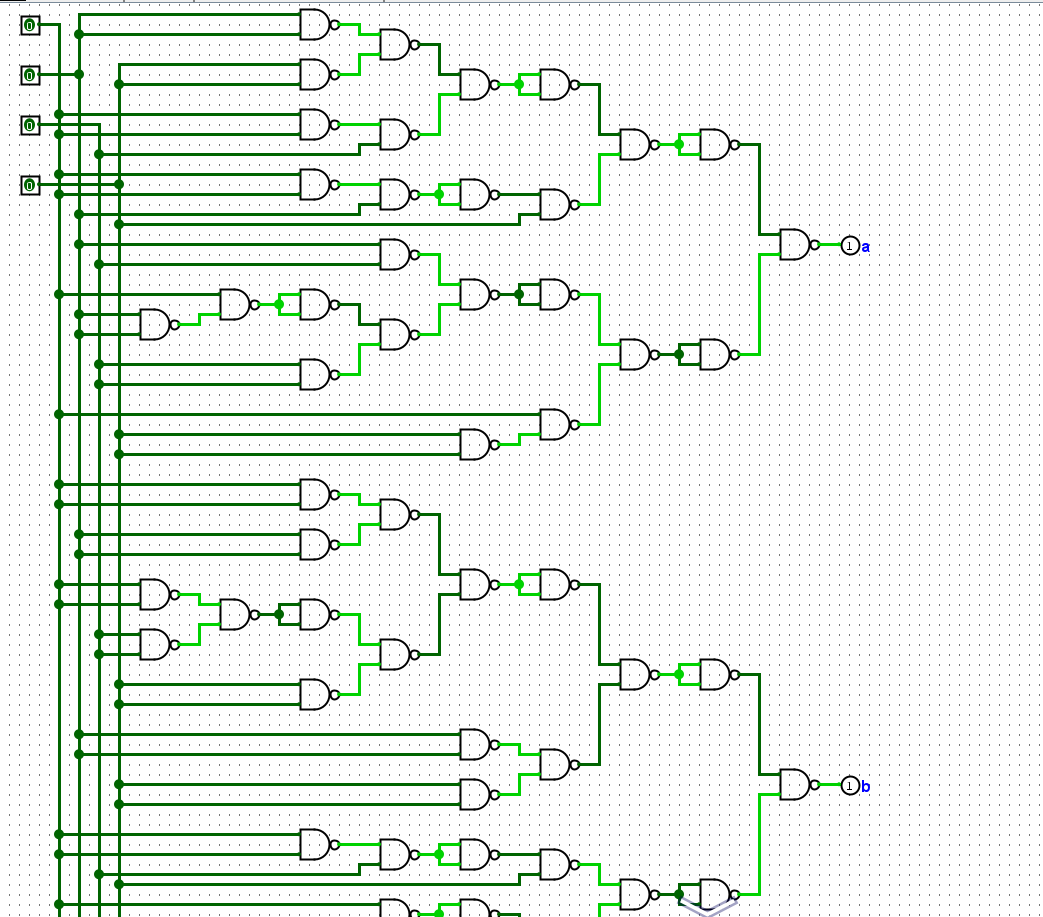
# Proceso de Diseño

## Sub Componente: Sub\_Seven\_Segment\_Decoder

Se diseña el decodificador de valores de 4 bits a 7 segmentos por medio de un circuito combinacional por medio de compuertas NAND. A este circuito se le da el nombre de **Sub\_Seven\_Segment\_Decoder** y hace parte del sub circuito diseñado para el laboratorio y que va inmerso en un componente mayor llamado **Binary\_to\_Two\_Seven\_Segment**.

### Análisis

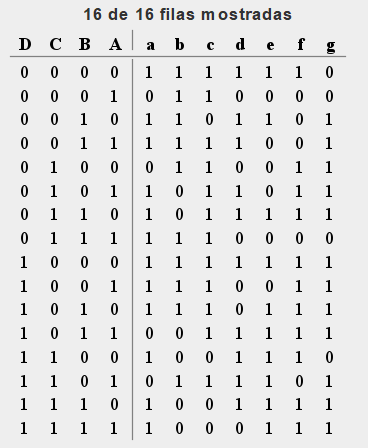
Se tiene como entrada los valores de 4 bits que luego son gestionados por compuertas lógicas para posteriormente dar una salida de 7 bits correspondientes a cada uno de los bits de los 7 segmentos. Para este se lleva a cabo un análisis combinacional que más adelante se describe.



### Analisis Combinacional

(D,C,B,A): 4 bits de entrada

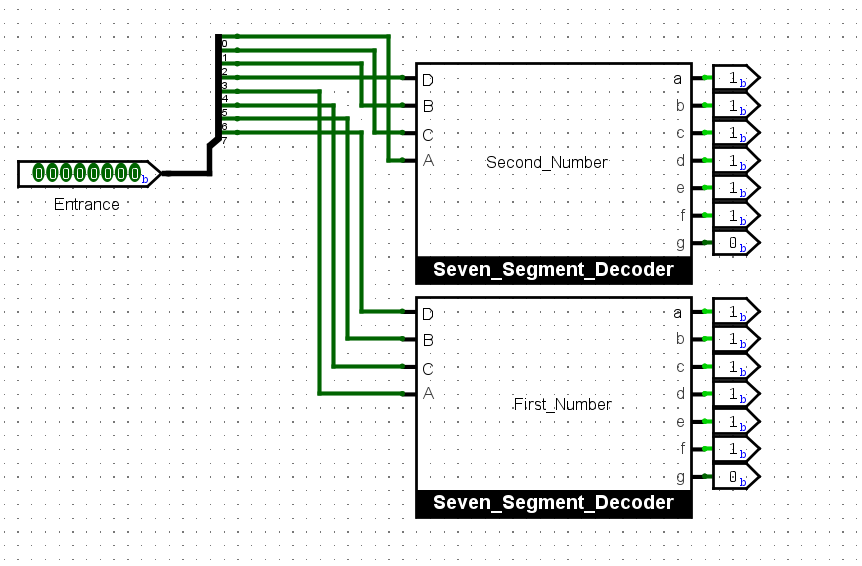
(a,b,c,d,e,f,g): 7 bits de salida que van directo al display de 7 segmentos



## Componente: Binary\_to\_Two\_Seven\_Segment

Se diseña el circuito que permitirá convertir una entrada binaria a un valor numérico decimal que se pueda visualizar en el display de 7 segmentos. Para este circuito se hace uso del sub circuito diseñado anteriormente de forma inclusiva e instanciada para que complemente la funcionalidad del presente circuito. Se usa el Seven\_Segment\_Decoder 2 veces para los 2 displays de 7 segmentos correspondientes a los 2 números que se mostrarán.

### Análisis



Se tiene una entrada en base binaria de 8 bits, la cual se separa por medio de un separador de 8 bits de entrada y 8 bits de salida, de los cuales los 4 primeros bits hacen referencia al segundo número que luego será decodificado y mostrado de forma decimal en el display de 7 segementos. Los 4 bits restantes de salida del separador, hacen referencia al primer número que luego será decodificado y mostrado de forma decimal en el display de 7 segementos.

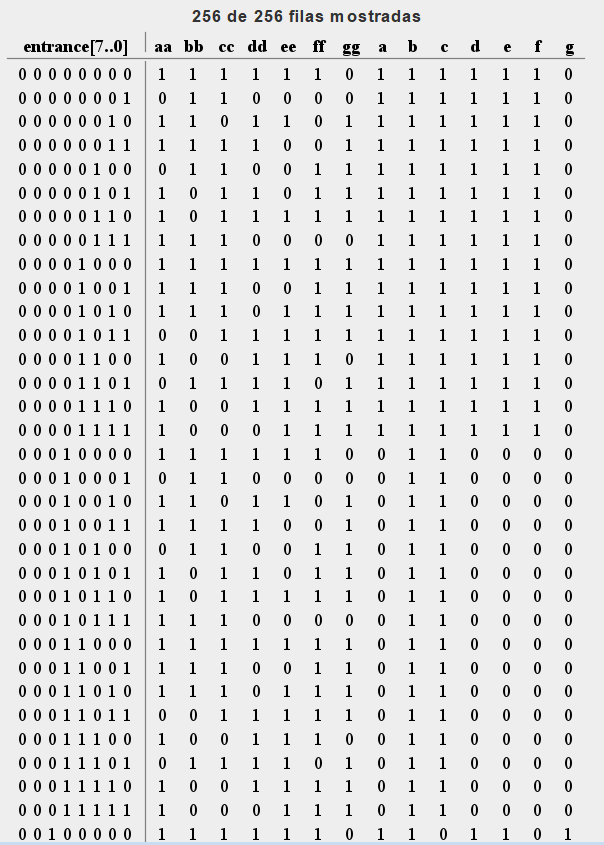
En el Seven Segment Decoder se evidencian los 4 bits de entradas y los 7 bits de salidas que conectarán directamente con el display de 7 segmentos. El Seven Segment Decoder es el mismo circuito diseñado anteriormente y que dimos por nombre: Sub\_Seven\_Segment\_Decoder

### Análisis Combinacional

(entrance[7..0]) : equivale a los 8 bits de entrada

(aa,bb,cc,dd,ee,ff,gg): equivale a los 7 bits de salida del display 7 segementos del número 2

(a,b,c,d,e,f,g): equivale a los 7 bits de salida del display 7 segementos del número 1



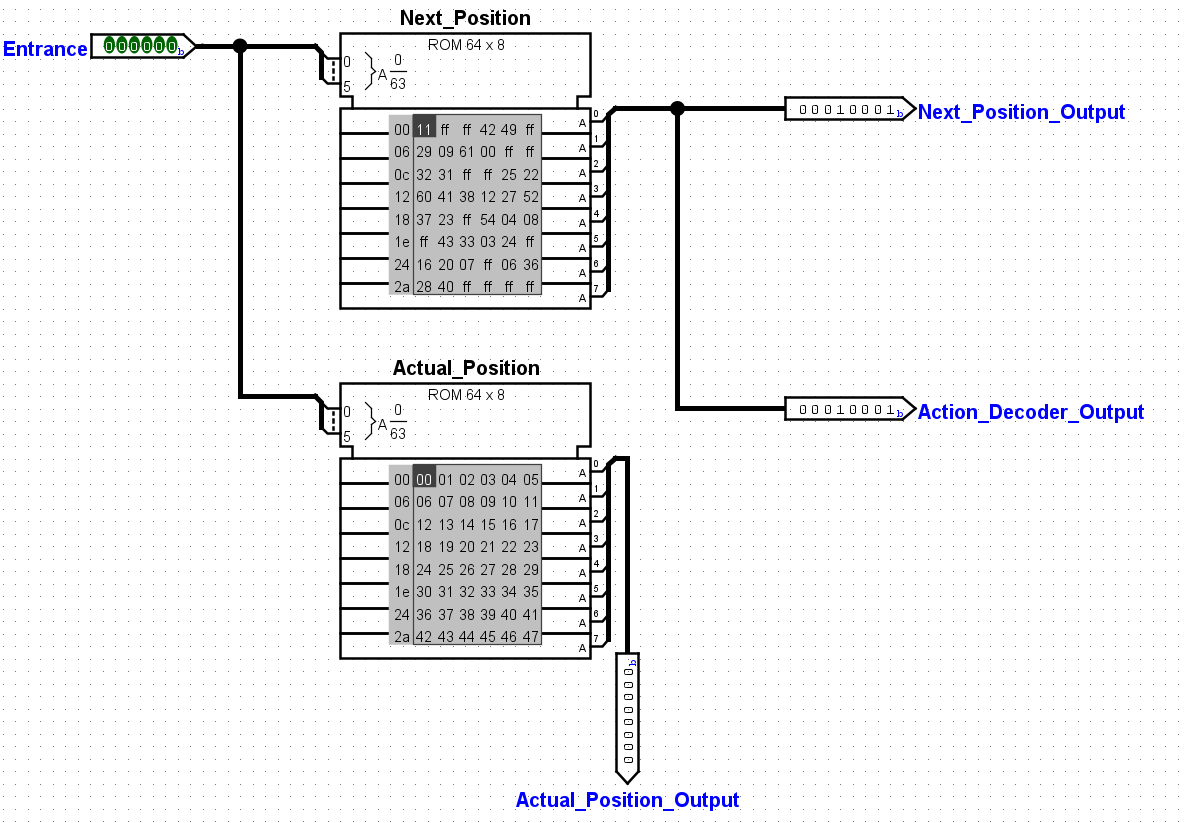
Doble clic para ver el análisis completo:



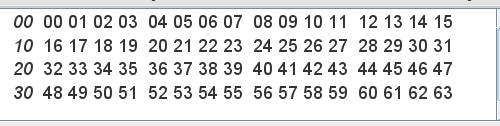
## Component: Movement\_Control

Componente encargado de generar la posición actual, la posición siguiente y la acción a realizar del robot de acuerdo a un número que entre en base binaria de 6 bits.

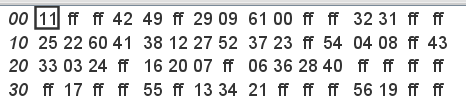
### Análisis



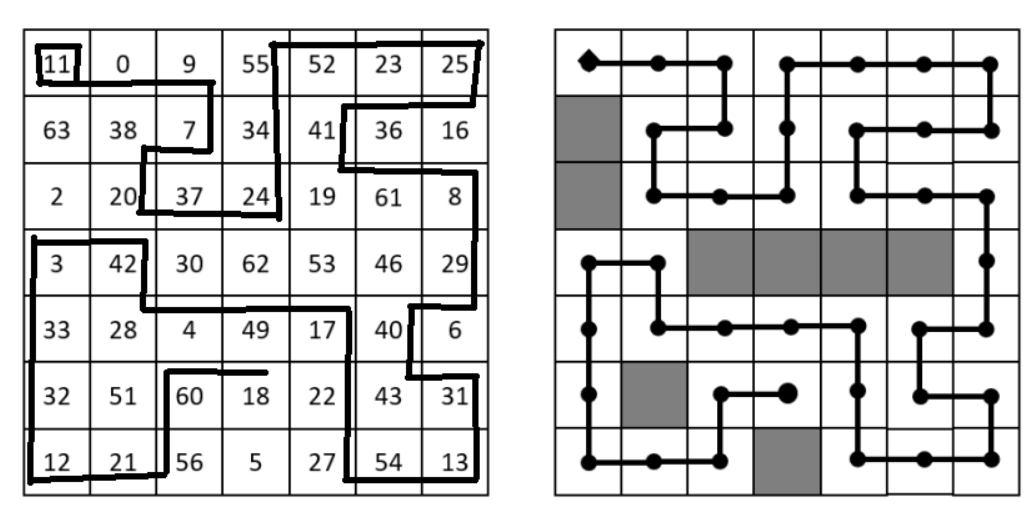
Se tiene una entrada binaria de 6 bits que conecta a 2 memorias ROM con entradas de 6 bits y salidas de 8 bits binarias. En la memorias ROM se mapean los respectivos valores de acuerdo a su posición. Para la ROM de posición actual se tienen los valores correctamente ordenados así:



Para la ROM de siguiente posición, se ubican los valores correspondientes a la siguiente posición de acuerdo a la posición actual.



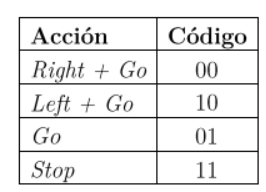
Por ejemplo si la posición actual es 0, la siguiente sería 11 de acuerdo a la tabla anterior. La tabla se crea partiendo del siguiente problema:



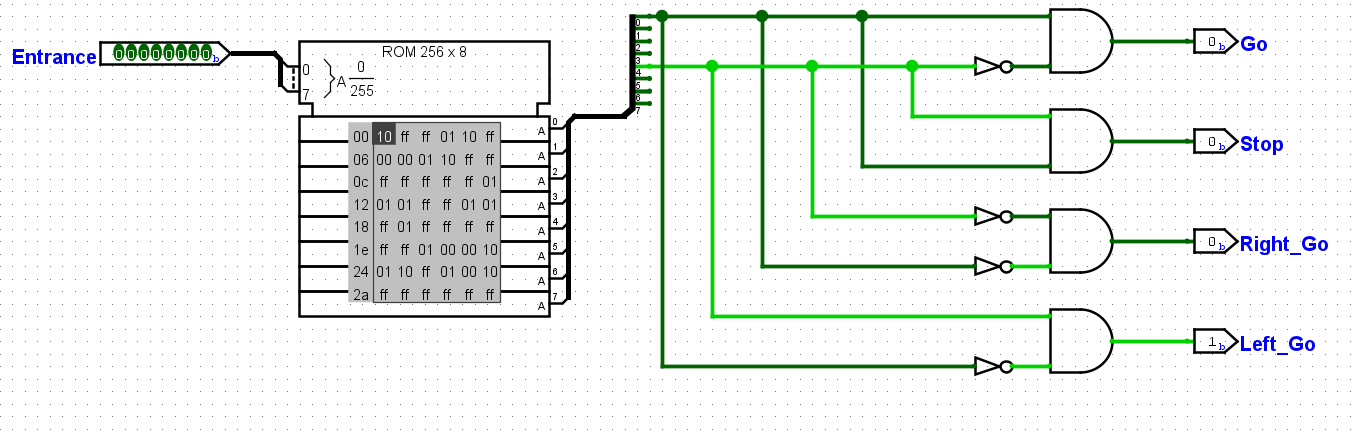
Los valores de FF son los que no se encuentran en la tabla o que están sombreados. Para el componente de Action\_Decoder se usa la salida de 8 bits de la ROM de Next\_Position.

## Componente: Action\_Decoder

Este es el componente encargado de decodificar la acción siguiente que tiene que realizar el robot para ir a la siguiente posición, esta acción se lleva a cabo de acuerdo al siguiente código del problema #2:

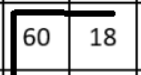


### Análisis

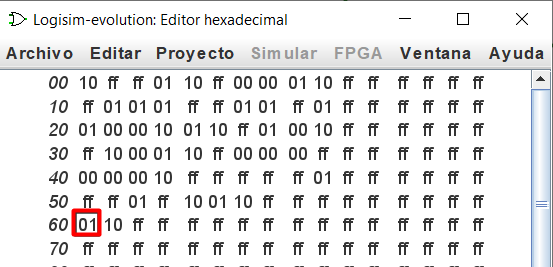


Se implementa una memoria ROM con valores correspondientes de 8x8, 8 bits de entrada y 8 Bits de salida. Los bits de entrada son provenientes del componente **Movement\_Control.**

Si el next\_position es por ejemplo 60:

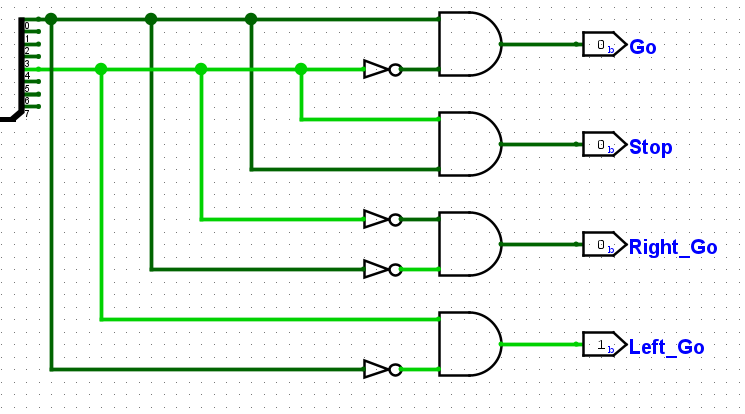


para ir de 18 a 60 debo de realizar un Go. Que según el código es el 01. entonces como dato de entrada llegará el 60 en binario, y el 60 en la memoria ROM tendrá valor de 01 así:



Se usa separador de bits para separar los 8 bits de salidas de la ROM, adicional se usan los bits 0 y 3 del separador como datos de entradas del circuito combinacional.

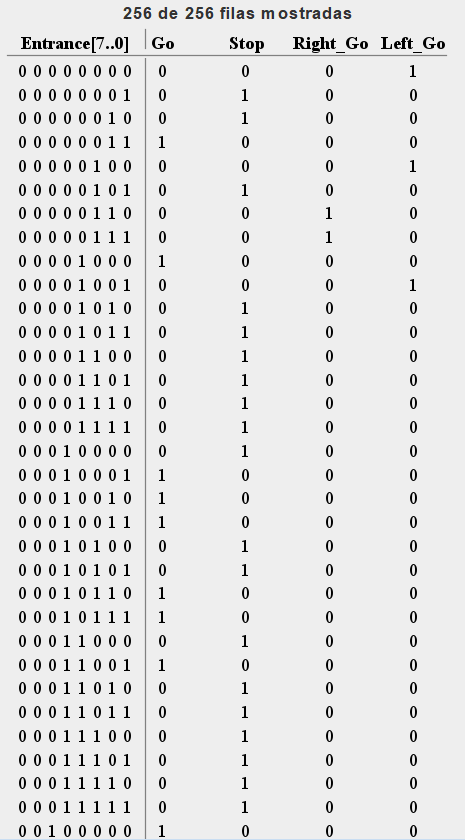
Se usan puertas AND y NOT para que cumpla con la lógica de acción de acuerdo a los códigos propuestos.



### Análisis Combinacional

(entrance[7..0]): equivale a los 8 bits de entrada

(Go, Stop, Right\_Go, Left\_Go): son cada una de las salidas del circuito y equivalen a 1 bit cada uno para que posteriormente sean conectadas a sus respectivos LED.

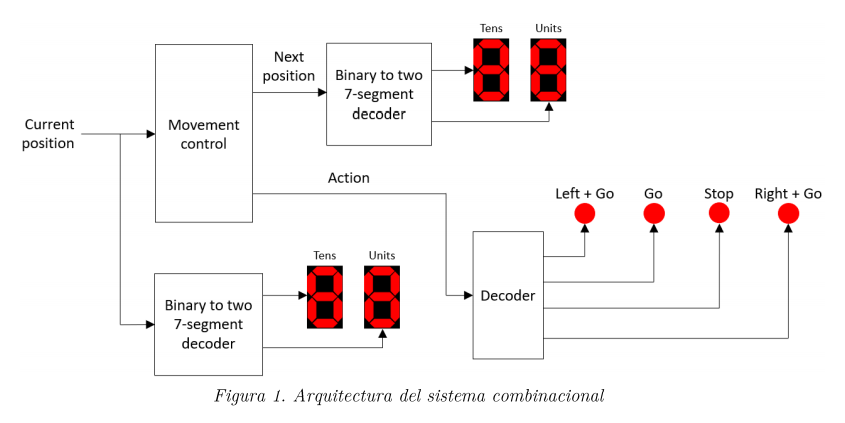


Doble clic para ver el análisis completo:

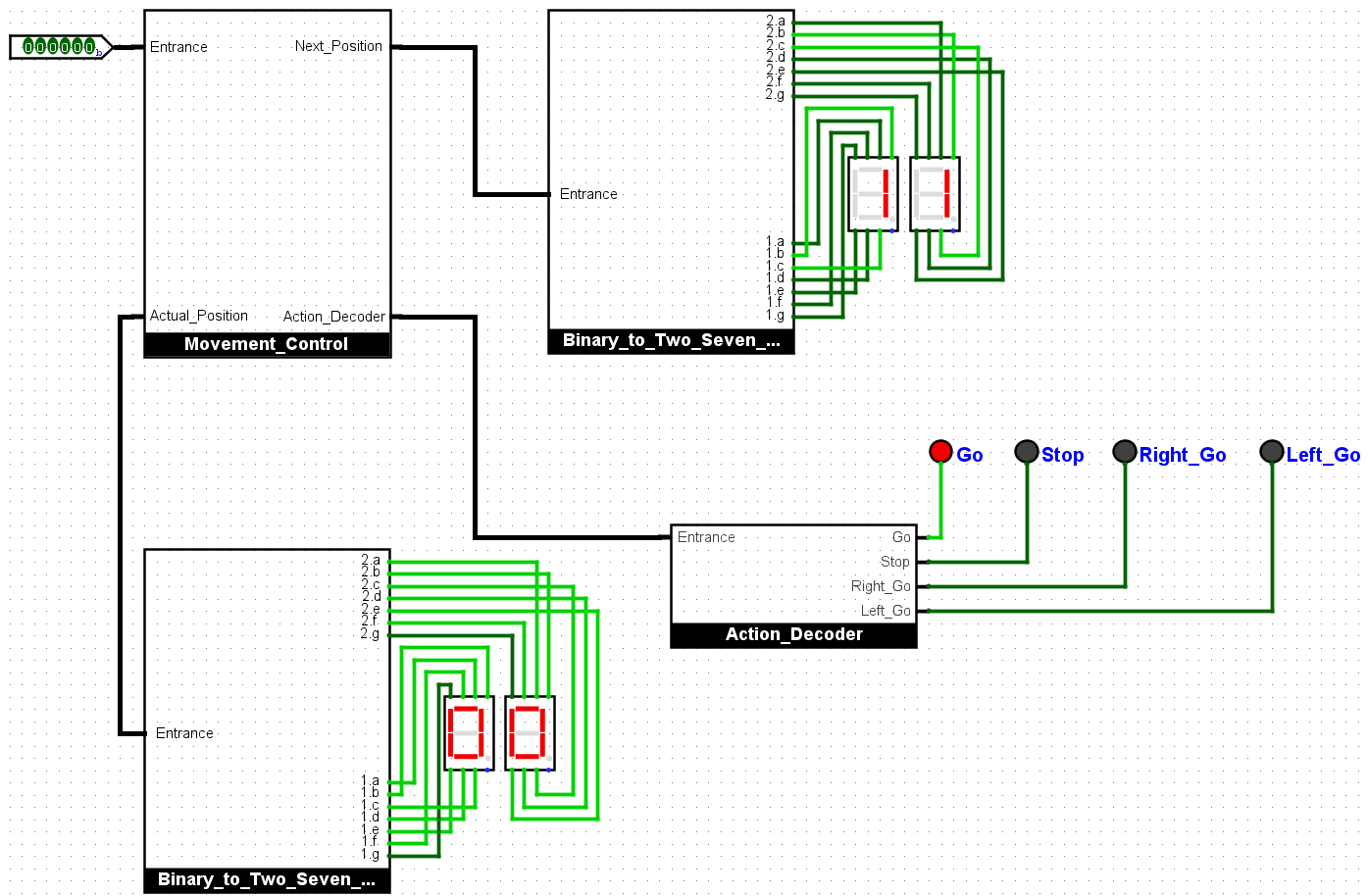


## Componente: Main

Este es el componente principal y dónde se integrarán los demás componentes mencionados anteriormente. Para el diseño de este componente se ha tenido en cuenta la arquitectura propuesta del circuito:



Componente Main diseñado:



### Análisis

Dicho circuito tiene una entrada binaria de 6 bits que conecta directamente al componente de **Movement\_Control** y este en si conecta los 2 componentes de **Binary\_to\_Two\_Seven\_Segment** y **Action\_Decoder.** Los cuales están conectados a displays de 7 segmentos y Leds. La comunicación entre componentes se realiza en 8 bits y sus salidas son de a 1 bits.

# Decisiones de Diseño

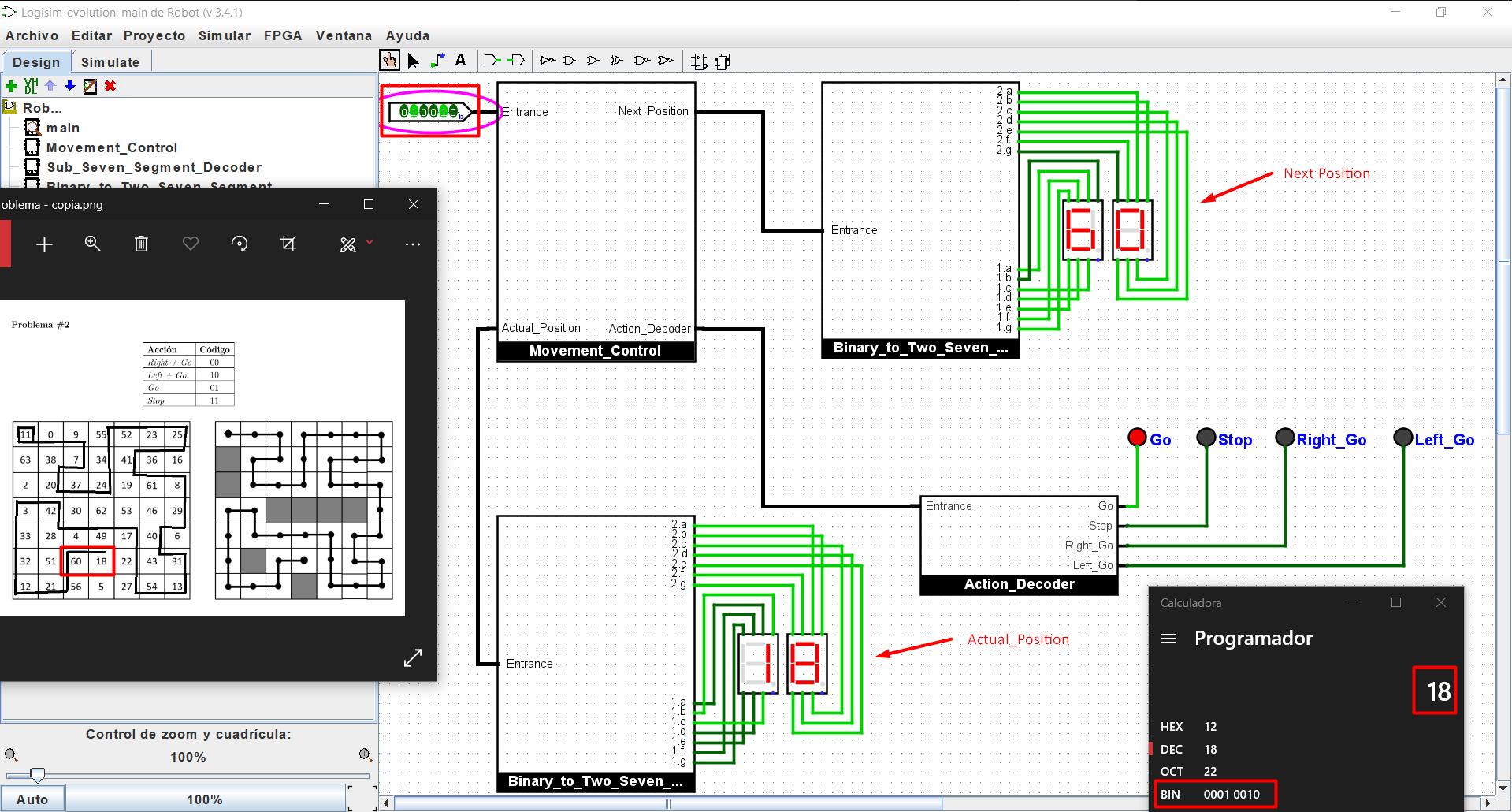
La posición de entrada es definida desde el componente de Movement\_Control y no directamente desde la entrada, ya que por una fácil implementación los componentes de Binary\_to\_Two\_Seven\_Segment reciben entradas de 8 bits y no de 6, adicional se toma esta decisión debido a que la arquitectura interna del componente Movement\_Control está diseñada para generar valores de 8 bits en decimal y no hexadecimal.

Se toma la iniciativa de usar memorias ROM para dar una solución ágil y limpia al problema planteado.

# Simulación

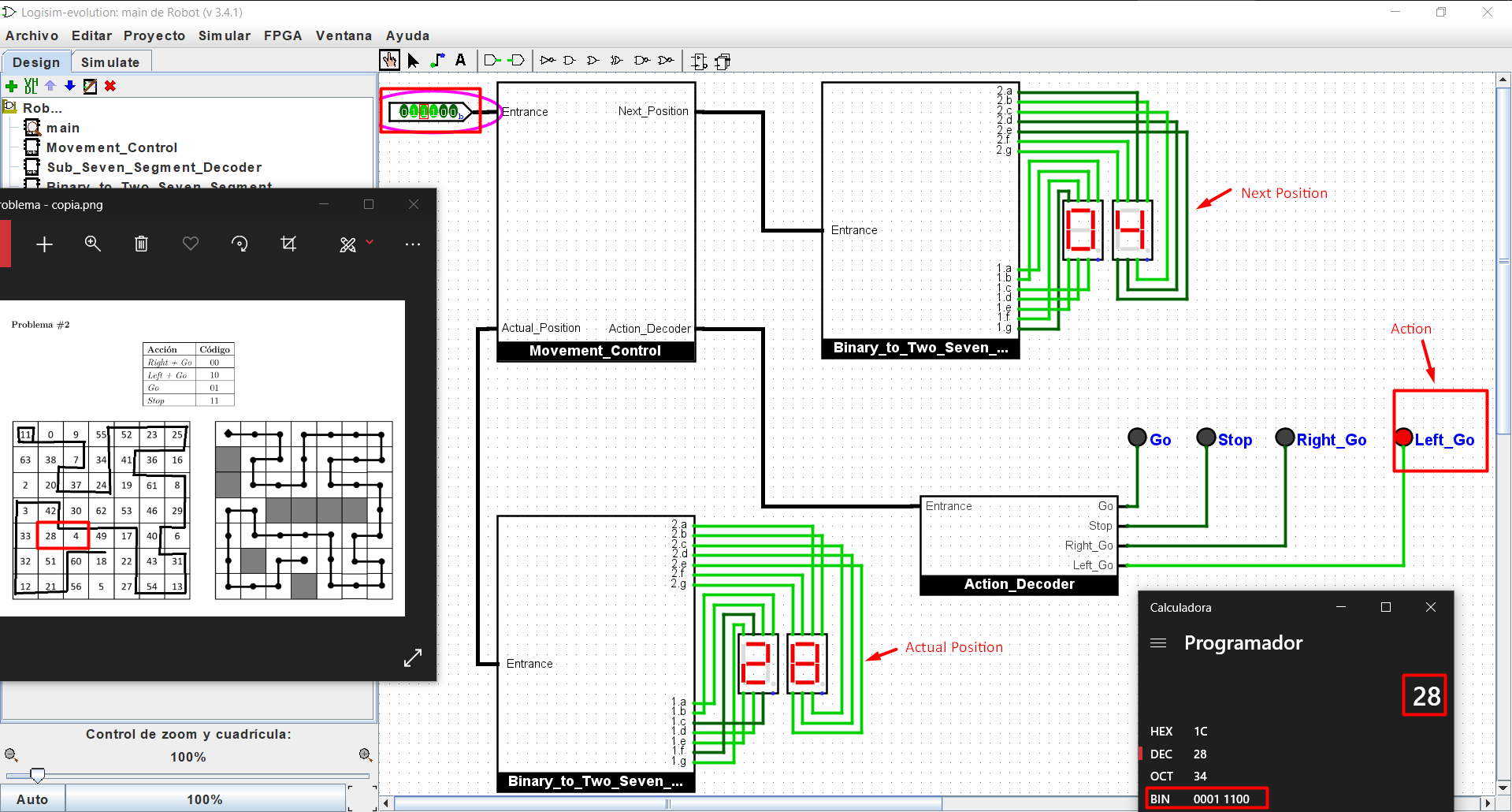
La simulación la haremos con base a distintos casos. Caso real, caso probable, caso extremos.

### Caso real 1:



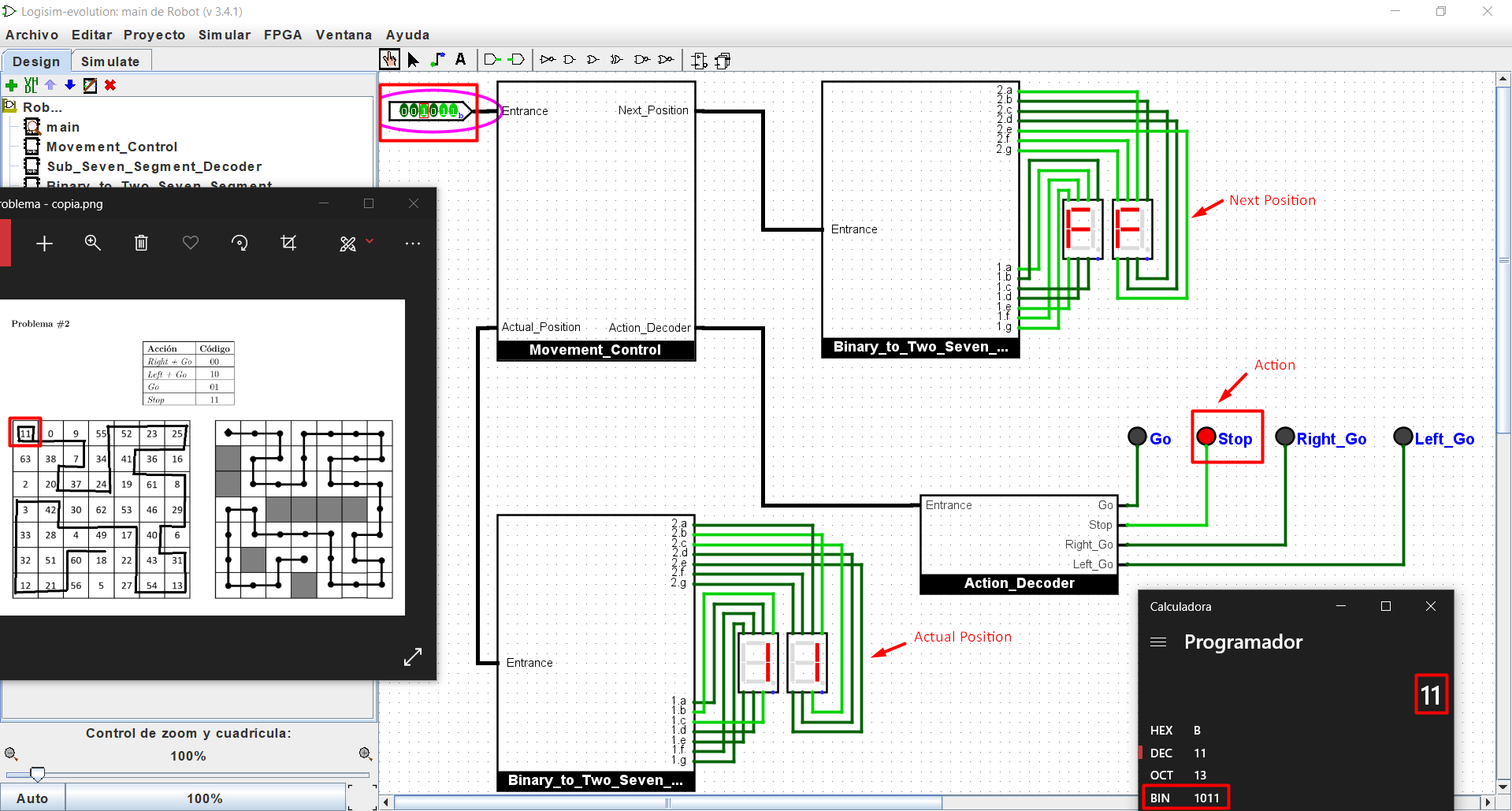
Visualizamos el número 18 en binario como: 010010 en la entrada y como posición actual, el LED indica que la acción es Go y que la siguiente posición es 60; de acuerdo al diagrama, es correcto.

### Caso Real 2:



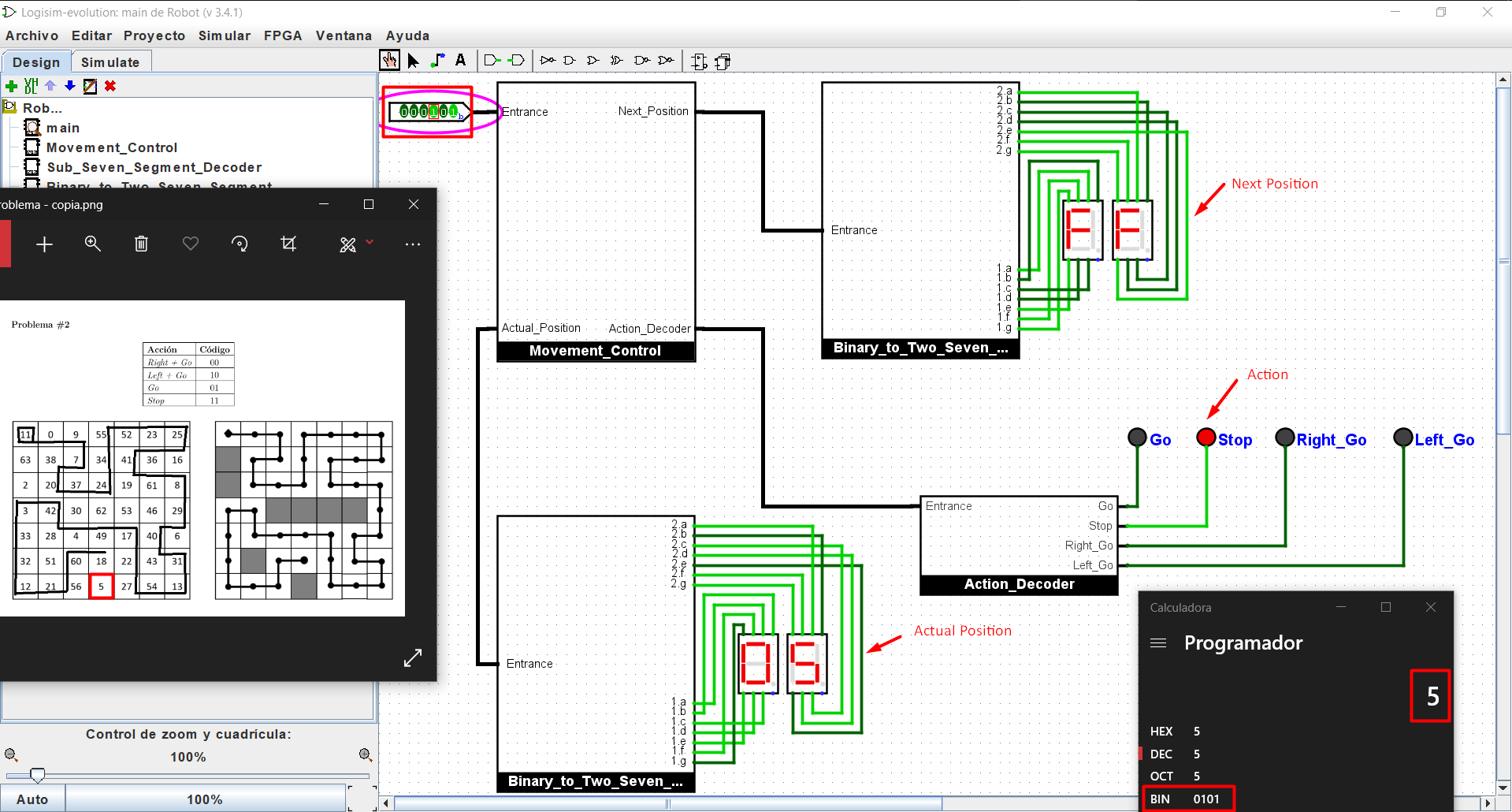
En este caso vemos que la posición actual es 28 denotada en binario como: 011100 de entrada, la acción a ejercer sería ir Left\_Go para ir a la posición siguiente que es 04, la cual se compara con el diagrama y muestra que es correcto.

### Caso Probable 1:



Este caso probable corresponde a que digiten como posición actual el 11. si vemos es el valor donde se quiere llegar y por consiguiente la acción a tomar es Stop y la siguiente posición es FF que corresponde a Fail.

### Caso Extremo 1:

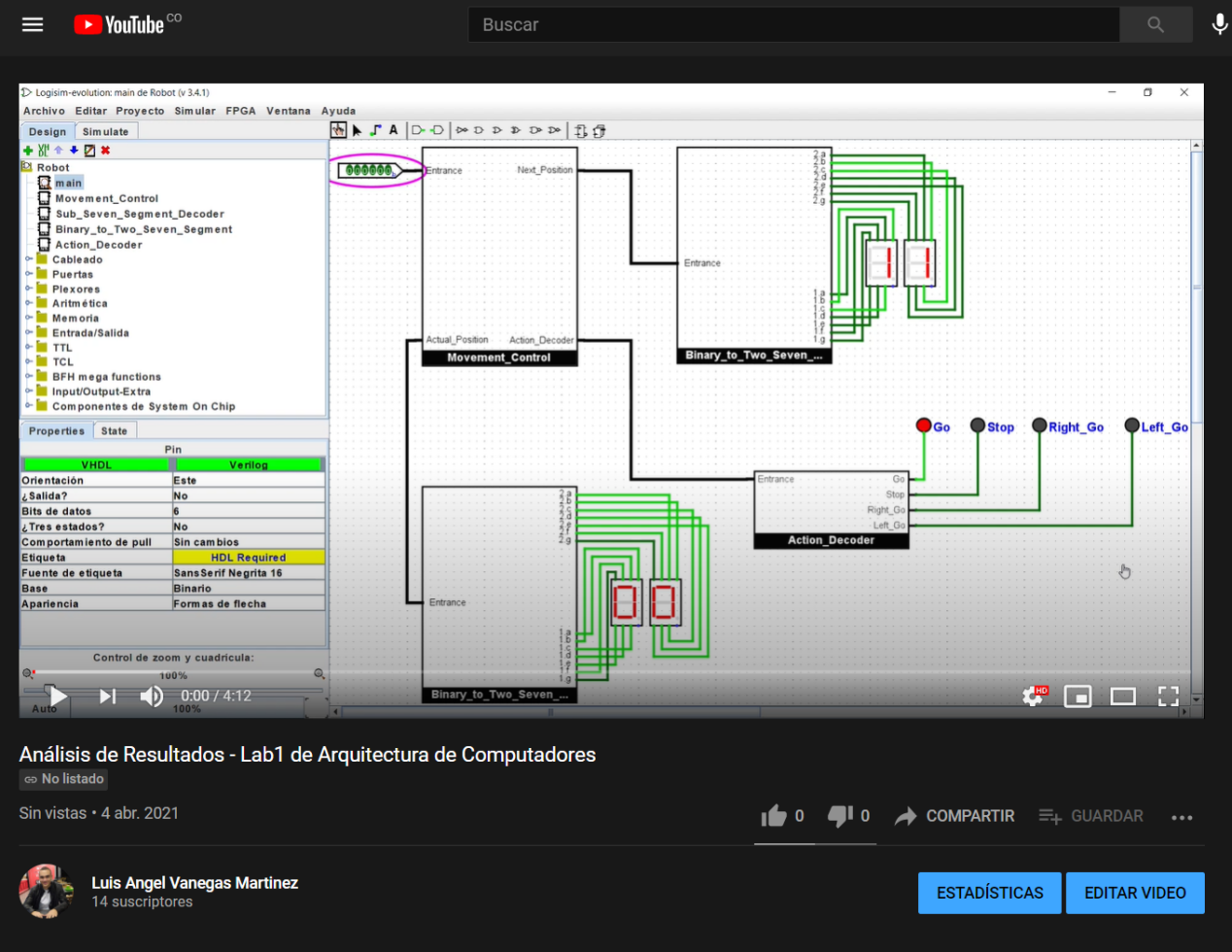


Este caso sucede cuando se pone una posición actual que no exista en la tabla o que este sombreada, mostrará el valor de entrada que se dio y la acción será Stop, adicional la siguiente posición es FF que corresponde a Fail.

# Análisis de resultados

Para un mejor entendimiento planteamos el análisis de resultados en forma de video para que vean la interacción de una forma dinámica y orientada a un caso de uso real.

Clic aquí para ver el vídeo: <https://youtu.be/9FAYKp8MSqY>



# Observaciones

El laboratorio está orientado a diferentes casos problemas, los cuales podrían aumentar en un futuro; la solución que he planteado tiene como objetivo suplir las necesidades de no sólo el problema #2 sino de cualquier otro problema propuesto, esto se haría sólo modificando los valores almacenados en la ROM del sistema y de resto, el resultado podría ser lo esperado.

Ejemplo: Si se requiere implementar la solución del problema #3 en el presente circuito digital, la arquitectura de este no se tocará; solo se modificará el archivo de memoria ROM con la nueva problemática y la respuesta será la esperada por el usuario.

# Conclusión

Para concluir, se hace uso de puertas lógicas y diseño de circuitos combinacionales para los componentes de bajo nivel como lo son: el Sub\_Seven\_Segment\_Decoder que se encarga de decodificar el valor binario de 4 bits que entra, en valores de 7 bits que pueden conectarse a un display de siete segmentos. También como el componente Action\_Decoder el cual se encarga de traducir los valores númericos de entrada en 8 bits a salidas de 1 bits que corresponde a acciones como Go, Left\_Go, Stop, Right\_Go.

Para los componentes de alto nivel usamos memorias ROM donde se mapean los valores de la tabla del problema propuesto, con entradas y salidas correspondientes a 8 bits y 6 bits (para las ROM de next\_position y actual\_position). Adicional hacemos usos de separadores de bits para que las salidas de los datos se puedan interpretar en bits unitarios y no en grupos de bits.

La solución esta pensada con el fin de favorecer una alta escalabilidad, con el fin de no amarrar el diseño a una solución sino crear un diseño que pueda satisfacer diferentes soluciones con tan solo un cambio que no sea de tan alto impacto y que interfiera con la arquitectura del circuito.

Repositorio del Trabajo: <https://github.com/xlavm/Logisim-Circuits>