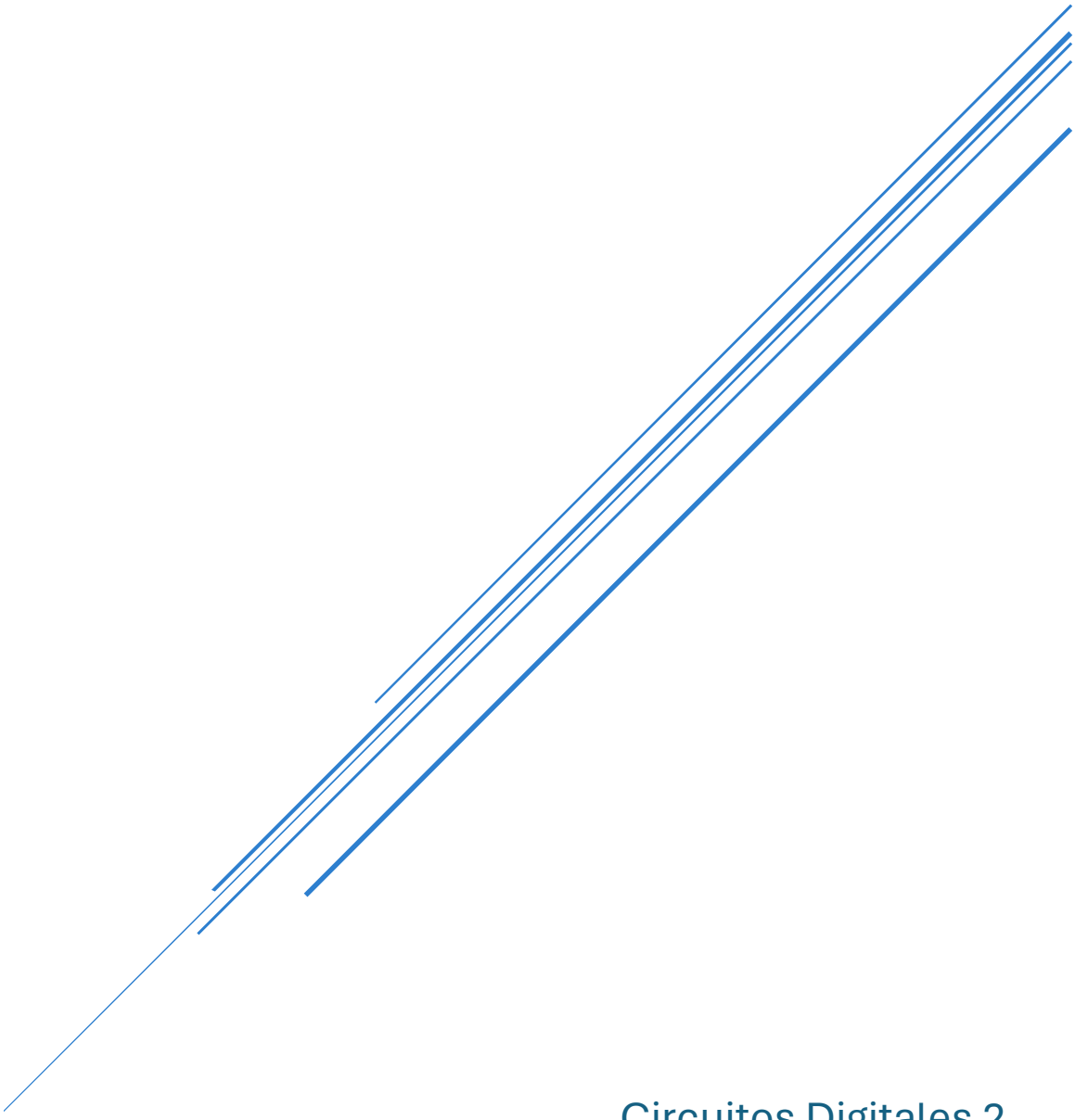
TAREA 1: REPORTE

MARÍA PAULETTE PÉREZ MONGE



Circuitos Digitales 2

13/4/2023

Resumen

En el siguiente reporte se presenta el diseño del controlador de la compuerta de un estacionamiento que se abre con contraseña, encendiendo una alarma si es incorrecta 3 veces. Este cuenta con diseño del circuito, un plan de pruebas para verificar su funcionamiento, así como la característica particular de un botón de enter para saber cuándo debe aceptar la contraseña de ingreso, el cual evita que, si el pin se mantiene activado durante varios ciclos, se cuente como contraseñas incorrectas para encender la alarma. Se realizaron pruebas tales como ingresar el pin más o menos de 3 veces, las cuales fueron exitosas y cumplen con lo solicitado. Se logró que la variable interna de contador de pines incorrectos se borrara en el mismo instante que se abre la compuerta. A partir de los resultados obtenidos fue posible predecir el comportamiento que tendrá el circuito cuando una entrada cambie justo en un flanco positivo de reloj, lo cual es importante de tomar en cuenta dentro del diseño.

- Parte 1: Descripción Arquitectónica:

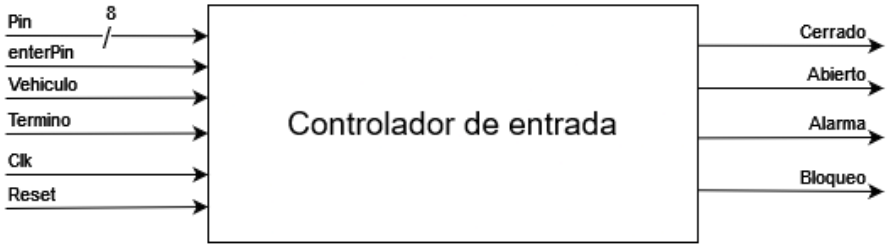


Figura 1: Diagrama de bloque del controlador.

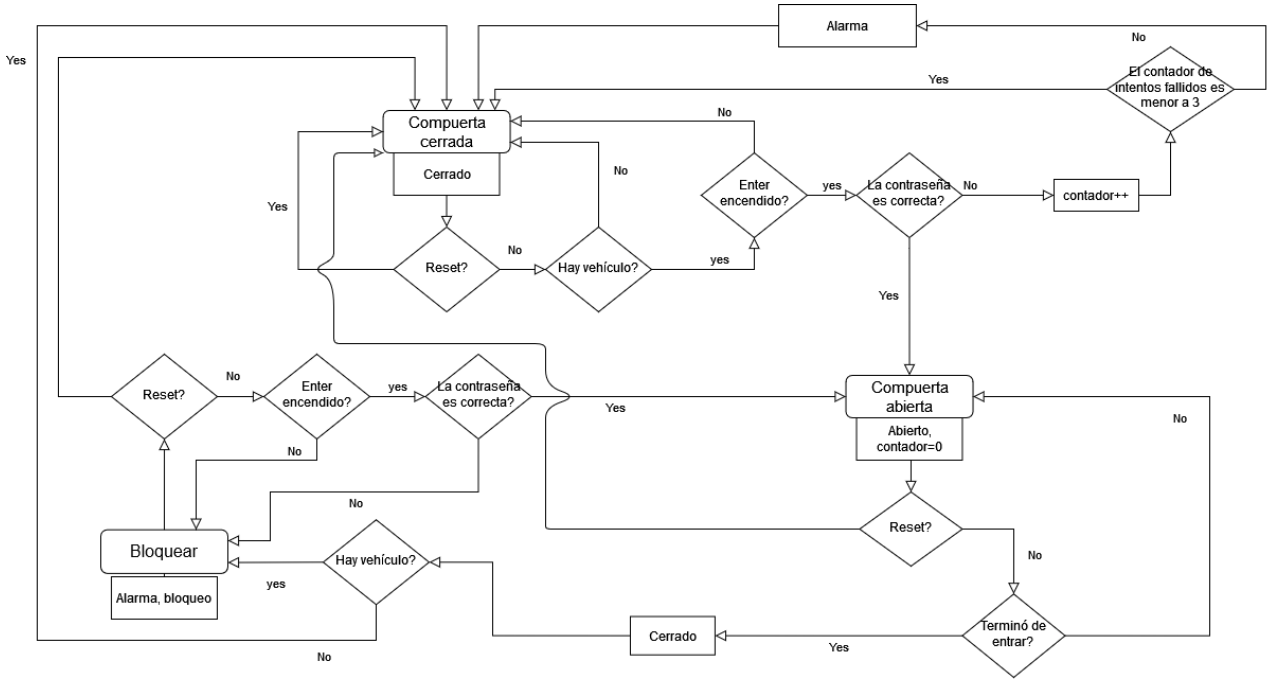


Figura 2: Diagrama ASM del controlador

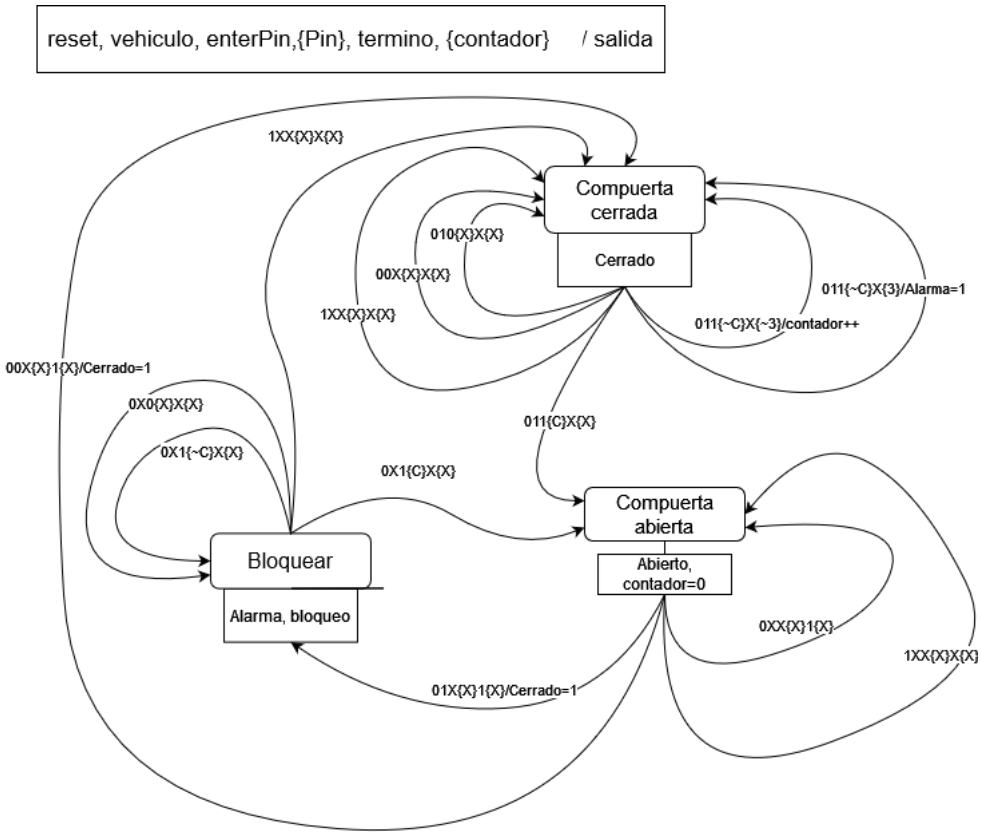


Figura 3: Diagrama de estados

En la figura 1 se observa que el controlador tendrá 6 entradas: un bus de 8 bits para introducir el pin, un botón de enter para que el pin se reconozca cuando este se aprieta, el sensor del vehículo, el sensor de que el vehículo haya terminado de pasar por la compuerta, el clock, el cual es necesario para toda máquina de estado, y el reset, que resetea la máquina. A su vez, el controlador tendrá 5 salidas, de las cuales, 4 irán a la compuerta para indicarle qué hacer, mientras que la salida de alarma se conectará a una alarma.

En la figura 2 es un diagrama hecho como guía para confeccionar el controlador, este incorpora las siguientes indicaciones proporcionadas por el enunciado. Empezaríamos en un estado de compuerta cerrada, con su respectiva señal de cerrar para la compuerta. Este estado es al que se regresaría al poner un 1 en la señal de Reset, sin importar el estado presente. Estando cerrada la compuerta, si un vehículo es detectado (a), se solicita la contraseña (b)**.** Si es incorrecta, la puerta sigue en el estado cerrado (d). Con **otra** variable interna, contaremos los intentos fallidos para ingresar la clave, y si este es el tercero, también volveremos al estado cerrado, pero con una salida condicional de alarma (e); en el código, cuando se llega a 3 no se cuenta más, pues es irrelevante cuantos han sido para este punto. En cambio, si la clave es correcta, pasamos al estado de compuerta abierta (c) y se limpia el contador de intentos fallidos (f). El pin sólo se recibirá si se prende un botón llamado enterPin durante un ciclo de reloj, esto para evitar que se aumente el contador si la señal de pin se deja encendida. Estando en el estado de compuerta abierta, analizamos la señal dada por el sensor que indica cuando el vehículo atravesó la puerta completamente (g) (<Termino>, para acortar). Si este indica que

ya terminó, se activa la señal de cerrar (esto aunque no estemos en el estado cerrar, pues esta acción, físicamente, dura un tiempo en ejecutarse) (h). justo después, pueden suceder 2 cosas: si el primer sensor, que detecta que otro vehículo quiere entrar, se activa mientras otro acaba de terminar de entrar, se pasará al estado de bloqueado y se encenderá una alarma (i) hasta que alguien digite la clave correcta encendiendo la señal de enter, lo cual provocará que la compuerta se abra de nuevo. Esto último no venía en los requerimientos, pero se hizo así para que el desbloqueo no tenga los mismos efectos que reset.

En la figura 3 se presenta el diagrama de estados. Cabe destacar que, en la esquina superior izquierda se señalan tanto las variables internas como las entradas, así como que el contador. C significa el pin correcto. En este caso, el pin correcto es 00010000, que equivale a 16 en decimal. Utilizando la codificación one hot, la asignación de estados es la siguiente:

- Cerrado=001
- Abierto=010
- Bloqueado=100

- Parte 2: Plan de pruebas

Pruebas	Tiempo	Pasó?
1. Prueba #1, funcionamiento normal básico. Llegada de un vehículo, ingreso del pin correcto y apertura de puerta, sensor de fin de entrada y cierre de compuerta.	0-85s	✓
2. Prueba #2, ingreso de pin incorrecto 3 o más veces. Revisión de alarma de pin incorrecto. Revisión de contador de intentos incorrectos. Ingreso de pin correcto, funcionamiento normal básico. Revisión de limpieza de contadores y alarmas.	85s-215s	✓
3. Prueba #3, ingreso de pin incorrecto menos de 3 veces. Llegada de un vehículo, ingreso de pin incorrecto (**una** o dos veces), puerta permanece cerrada. Ingreso de pin correcto, funcionamiento normal básico. Revisión de contador de intentos incorrectos.	215s-265s	✓
4. Prueba #4, alarma de bloqueo. Ambos sensores encienden al mismo tiempo, encendido de alarma de bloqueo**, ingreso de clave incorrecta, bloqueo permanece.** Ingreso de clave correcta, desbloqueo. Funcionamiento normal básico.	265s-335s	✓
5. Prueba #5, botón de enter. En el estado cerrado, pin cambia sin el botón apretado tanto a uno incorrecto como al correcto, apretamos enter para abrir la compuerta, Verificamos que el contador de intentos fallidos no crezca. Bloqueamos la compuerta, pin cambia sin el botón apretado, tanto a uno incorrecto como al correcto, apretamos enter. Volvemos al estado abierto.	335s-485s	✓
6. Prueba #6, reset. De la prueba anterior estamos en el estado abierto, encendemos vehículo y termino para ir al estado bloqueado, las apagamos, pues no va a cambiar el estado que estén en 0. Ponemos todas las entradas en 1. Revisamos que reset tenga prioridad. Ya en el estado cerrado, ponemos todas las entradas en 1. Revisamos que reset tenga prioridad. Encendemos el sensor de vehículo e introducimos el pin	485s-655s	✓

correcto para abrir la compuerta. Ponemos todas las entradas en 1. Revisamos que reset tenga prioridad.		

- Parte 3: Instrucciones de uso

Para efectuar las pruebas propuestas, es necesario ir hacia la carpeta donde están guardados los archivos este proyecto mediante la terminal de su sistema Linux, ingresando el comando

$cd <dirección de la carpeta>

Posteriormente, ingresar el siguiente comando:

$make

Este último ejecutará el Makefile el cual, a su vez, ejecutará los comandos dentro de este.

- Parte 4: Ejemplos de los resultados

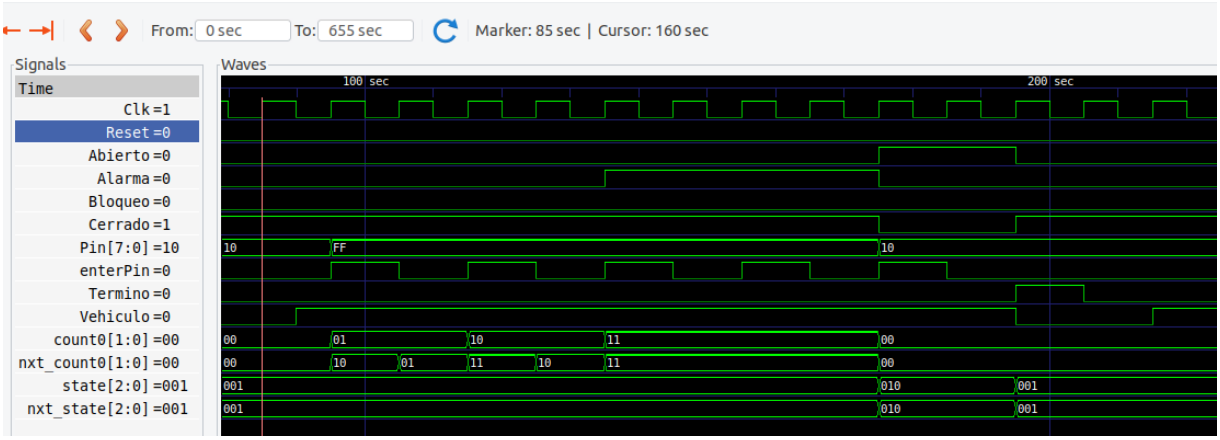


Figura 4: Ejemplo 1, prueba 2.

Uno de los ejemplos a analizar será la prueba 2 al ser representativa del proyecto. Según la figura 4, el vehículo se hace presente en el segundo 90 mientras que la contraseña se introduce en el 95. Es importante resaltar que nxt_count pasó de 0 a 2 justo en ese mismo flanco de reloj. Después de analizar los resultados, la explicación a la que se llegó es que cuando las entradas cambiaron, la lógica combinacional actuó primero y sumó 1 al nxt_count. Inmediatamente después, la lógica secuencial colocó el contenido de nxt_count en count. Debido a este cambio en una variable interna, la lógica combinacional se activa de nuevo y se vuelve a sumar 1 en nxt_count justo en el instante posterior al flanco creciente, sin embargo, su valor no se pasa a count y más bien es reescrito con el valor de este porque en el segundo 105 se apaga enterPin, lo cual activa la lógica combinacional antes que la secuencial. Esta contiene una línea que reescribe el valor actual de count en nxt_count antes de evaluar el estado presente, como se observa en la figura 5:

```
always @(*) begin
  //por defecto
  nxt_state = state;
  nxt_count0 = count0;

  case (state)
    C_Cerrada: begin //si empezamos
```

Figura 5: Extracto de código del bloque combinacional

De estos resultados podemos inferir que, cuando una señal cambia al inicio de un flanco de reloj, la lógica combinacional se activa primero que la secuencial. Esto fue útil para decidir la forma en la que el contador funcionaría, pues podemos hacerlo llegar a 3 y hacer que se prenda la alarma cuando justo en el mismo instante, aunque esto requiera pasar de nuevo por la lógica combinacional, pues, al haber cambio en una variable interna, se activará otra vez. Al inicio, desconociendo este comportamiento, se había optado por activar la alarma si el contador estaba en 2 y ese ingresaba un pin incorrecto, pero esto es más confuso de comprender en el código. Aunque esta cualidad dependa de que los cambios sean en el flanco de reloj, según el profesor, este es el comportamiento esperado en la vida real.

Otro detalle relevante de la figura 4 es que se pude apreciar que, aunque la contraseña incorrecta siempre esté en alto, no sucede nada si el pin no se aprieta. Esto fue muy útil, ya que, si hiciéramos la señal sólo con el pin, el contador aumentaría en cada ciclo de reloj.

Cabe destacar que, para lograr que el contador se borrara en el mismo instante que se pasa al estado abierto, se puso una asignación bloqueante de este valor, cada vez que el este era el estado próximo, como se observa en la figura 6:

```
if (Pin==Pin_correcto) begin
  nxt_state = C_Abierta; //si hay
  nxt_count0 = 2'b00; //Cuando se
end
```

Figura 6: Extracto de código con nxt_state= C_Abierta

Sin ésta, el contador se borraba un ciclo de reloj después de cambiar de estado, lo que no concuerda con el diseño y puede causar errores en la lógica combinacional durante este periodo.

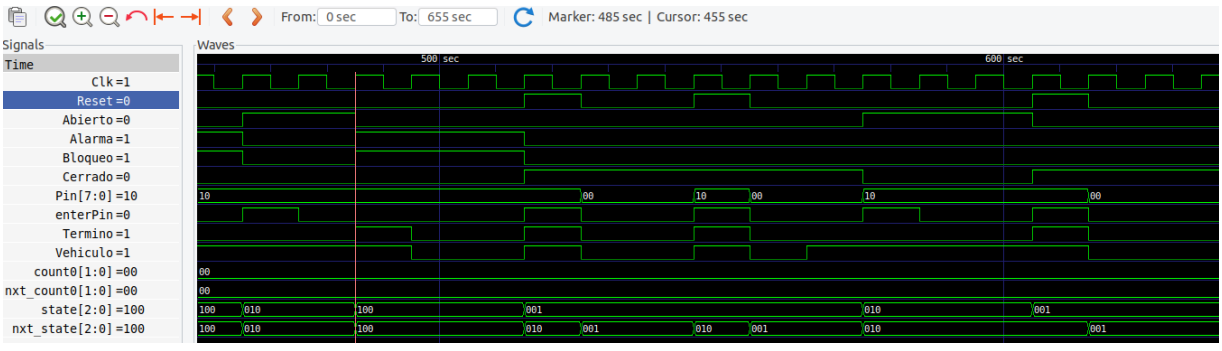


Figura 7: Ejemplo 2, prueba 6

Por otro lado, la figura 7 muestra que la prueba 6 inicia en el segundo 485 a partir de un estado abierto. Aprovechando que nos encontramos en este, encendemos la señal de vehículo y Termino para bloquear la puerta y pasar al estado bloqueado. En el segundo 515, encendemos todas las entradas. Como podemos observar, la lógica combinacional detecta primero, en el flanco creciente, que Vehiculo y Termino están en alto, lo que causa que ponga Abierto en nxt_state, mas este no se convierte en el actual porque un instante después, la lógica secuencial actúa y se detecta que reset está encendido, pasando al estado cerrado y reescribiendo nxt_state en el siguiente ciclo de reloj debido al código de la figura 5, donde las entradas se apagan. El mismo principio aplica para el segundo 545, cuando la puerta está cerrada y el 605 cuando pasa de abierta a cerrada.

- Parte 5: Conclusiones y recomendaciones

Conclusiones

- Se concluye que, después de implementados todos los elementos, la máquina de estados cumple con los requerimientos solicitados en el enunciado.
- Se determina que la lógica combinacional se activa primero que la secuencial cuando una señal cambia al inicio de un flanco de reloj.
- A su vez se concluye que, si la lógica secuencial realiza algún cambio a las variables del sistema en el flanco de reloj, la lógica combinacional reaccionará justo en ese instante.
- Mientras no interfiera con las especificaciones de diseño, es posible añadir entradas o variables internas que le aporten un mejor funcionamiento.

Recomendaciones

- Si se utiliza una máquina virtual, es recomendable tener un backup del código, por ejemplo, Github, ya que es fácil que ésta presente errores o se corrompa y ya no se pueda volver a abrir.
- Se deben tener bases teóricas de cuáles son las condiciones en las cuales la máquina utiliza la lógica secuencial o la combinacional, ya que esto contribuye a un diseño correcto y al entendimiento de los diagramas de tiempo obtenidos.
- Para próximas pruebas, encender las señales antes de los flancos crecientes o a mitad de los periodos, para así reconocer distintos comportamientos a los esperados.