Instituto Tecnológico de Buenos Aires

Trabajo Práctico N°3
Electrónica III - 2019

Grupo 1:

Farall, Facundo David
Gaytan, Joaquín Oscar
Kammann, Lucas Agustín
Maselli, Carlos Javier

Profesores:

Dewald, Kevin
Wundes, Pablo
Aguirre, Miguel

13 de noviembre de 2019

Índice

Ejercicio 1: Control de bombas de agua

Se pretende implementar un sistema controlador de dos bombas de agua cuya función es mantener el nivel de agua de un tanque entre un rango marcado por dos sensores de nivel. Ambas bombas estarán encendidas cuando el nivel de agua esté por debajo del mínimo (sensor I=0), y apagadas cuando este supere el máximo (sensor S=1). En un nivel intermedio (I=1 y S=0), solo una de las bombas se encontrará trabajando, y aquella en hacerlo será la última en haber permanecido apagada mientras la otra bomba estaba en funcionamiento; es decir, si en determinado momento está trabajando solo la bomba 1 (B1), la próxima vez que se dé la condición I=1 y S=0, se pondrá en funcionamiento la bomba 2 (B2), y vice versa. Es en esta característica de m̈emoriaën donde el planteo de una solución a la problemática mediante una máquina de estados, se vuelve considerable.

Diseño de Máquina de Estados

Se propone como diseño para la máquina de estados aquella que sigue el diagrama de la Figura 1. Cabe destacar que para la combinación de entradas I=0 y S=1, que resulta imposible en la aplicación del circuito a la realidad, dado que el sensor I se encuentra por debajo del sensor S, se decidió apagar ambas bombas por seguridad.

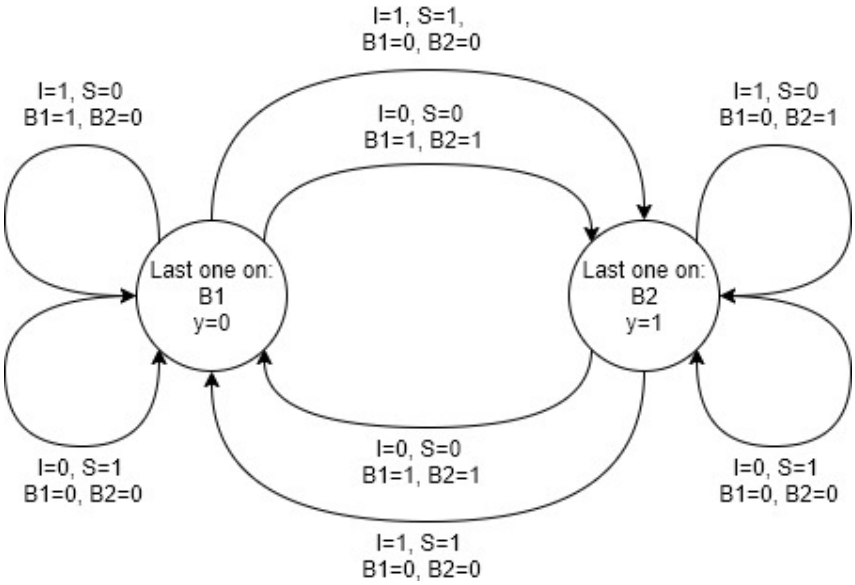

Figura 1: Diagrama de estados para sistema de control de bombas.

Como se puede observar, la máquina de estados planteada corresponde a una implementación de Mealy. La decisión de utilizar esta implementación por sobre una de Moore, es que una aplicación estricta de la segunda, donde la salida dependiera únicamente del estado actual, hubiera supuesto el uso de 6 estados, resultando altamente ineficiente en comparación a la solución elegida.

La Tabla 1 representa la tabla de verdad correspondiente a la máquina de estados en cuestión, donde ÿës el estado actual, Ïÿ S̈las entradas, Ÿël siguiente estado a partir del próximo clock, y B̈1ÿ B̈2l̈as salidas asincrónicas.

I	**S**	**y**	Y	B1	B2
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	0	0	0

Tabla 1: Tabla de verdad para máquina de estados de control de bombas.

De la observación de las salidas, se infiere que los circuitos lógicos para implementarlas son los de las Figuras 2, 3 y 4. Luego, la máquina de estados estará conformada por el circuito de la Figura 5.

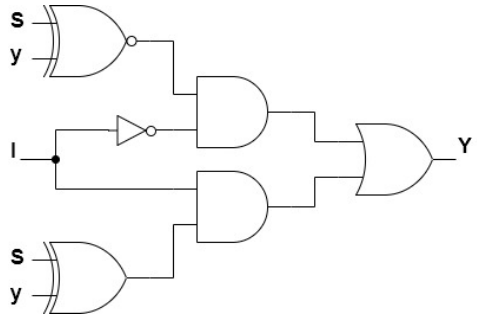

Figura 2: Circuito lógico para la entrada del Flip-Flop D.

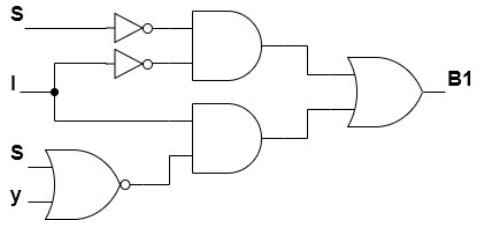

Figura 3: Circuito lógico para la salida de la bomba 1.

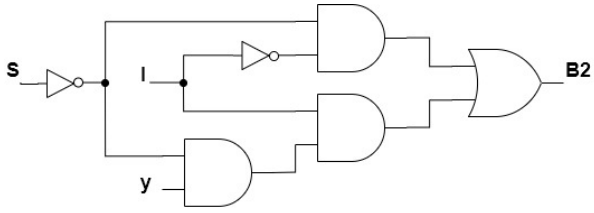

Figura 4: Circuito lógico para la salida de la bomba 2.

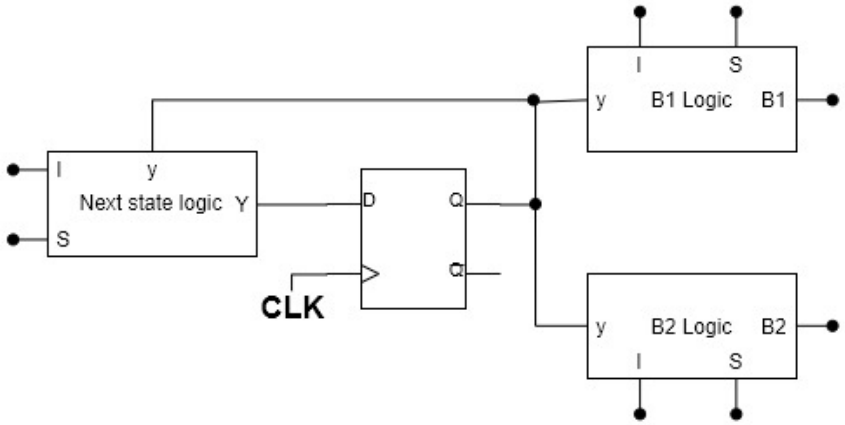

Figura 5: Circuito de la máquina de estados.

Simulaciones en Verilog

Se desea comprobar el funcionamiento de la máquina de estados a nivel lógico mediante una simulación en Verilog, para lo cual se emplean dos metodologías de diseño. Por un lado, puede utilizarse un diseño en Verilog que determine si la máquina está bien diagramada, utilizando para ello el bloque procedural case. Por otro lado, para determinar si la implementación lógica de la máquina puede funcionar, debe emplearse un diseño a nivel compuertas de los módulos en Verilog, para esto último se divide el problema inicialmente en tres bloques, los flip flops, la lógica combinacional que produce el próximo estado y la lógica de salida. Finalmente, un cuarto bloque o módulo describe la máquina interconectando los módulos mencionados para producir el comportamiento esperado.
En la Figura 6 se observa el comportamiento de la máquina de estados ante los diferentes escenarios de prueba. Se puede apreciar como el comportamiento de la cantidad de bombas que deben encenderse es el correcto, es decir, cuando los dos sensores mandan señal *low*, ambas bombas se encuentran encendidas, cuando, en cambio, lo que ingresa son dos señales *high*, ambas bombas se apagan. Cuando el agua se encuentra entre los dos sensores, la bomba que se enciende se va alternando, y, finalmente, ante la situación no esperada, se apagan las bombas por seguridad.

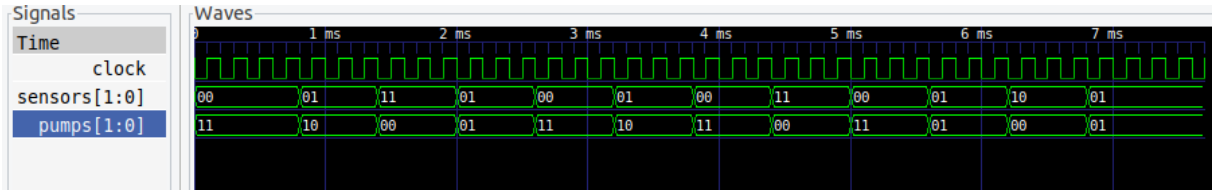

Figura 6: Salida de la simulación de la máquina de estados en GTKWave.

La misma simulación puede realizarse observando los resultados desde linea de comandos, y la salida es la siguiente:

```
Running simulation!
VCD info: dumpfile bin/output.vcd opened for output.
Testing I=0 S=0. Output: 11. Next state is 1. Current state is 0
Testing I=1 S=0. Output: 10. Next state is 1. Current state is 1
Testing I=1 S=1. Output: 00. Next state is 0. Current state is 1
Testing I=1 S=0. Output: 01. Next state is 0. Current state is 0
Testing I=0 S=0. Output: 11. Next state is 1. Current state is 0
Testing I=1 S=0. Output: 10. Next state is 1. Current state is 1
Testing I=0 S=0. Output: 11. Next state is 0. Current state is 1
Testing I=1 S=1. Output: 00. Next state is 1. Current state is 0
Testing I=0 S=0. Output: 11. Next state is 0. Current state is 1
Testing I=1 S=0. Output: 01. Next state is 0. Current state is 0
```

```
Testing impossible situation. Output: 00. Next state is 0. Current state is 0
Testing I=1 S=0. Output: 01. Next state is 0. Current state is 0
```

Diseño en PCB

Se procedió a realizar la implementación de la máquina de estados en un PCB, para el cual algunas modificaciones que favorecían el rendimiento de espacios y componentes fueron realizadas. Un ejemplo de estos cambios fue reemplazar las salidas de dos ANDS a una OR por MUXES, con la entrada I como selectora y las restantes entradas de las ANDS como entradas de los MUX.
El resultado fue el PCB de la Figura 7.

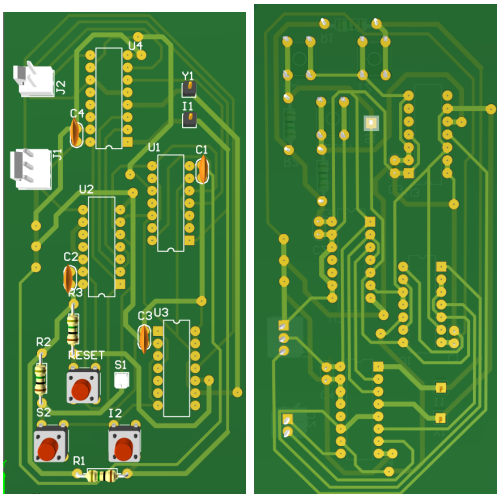

Figura 7: Implementación en PCB de la máquina de estados.

Resultados

El circuito lógico que implementa la máquina de estados fue sometido a diferentes estímulos que simulaban, mediante pulsadores, el accionar de los sensores del tanque. Se midieron las salidas que comandarían las bombas y se comprobó que su comportamiento fuese el esperado.
Los resultados son los que se muestran en la Figura 8, en la cual la imagen mde la izquierda es la salida inalterada, mientras que a la derecha se remarcan sobre los mismos datos obtenidos del osciloscopio, las diferentes transiciones en la salida. El canal 1 en ambas imágenes es la señal de CLOCK (amarillo), mientras que el canal 2 (verde) y 3 (violeta) son la salida de la bomba 1 y 2, respectivamente.

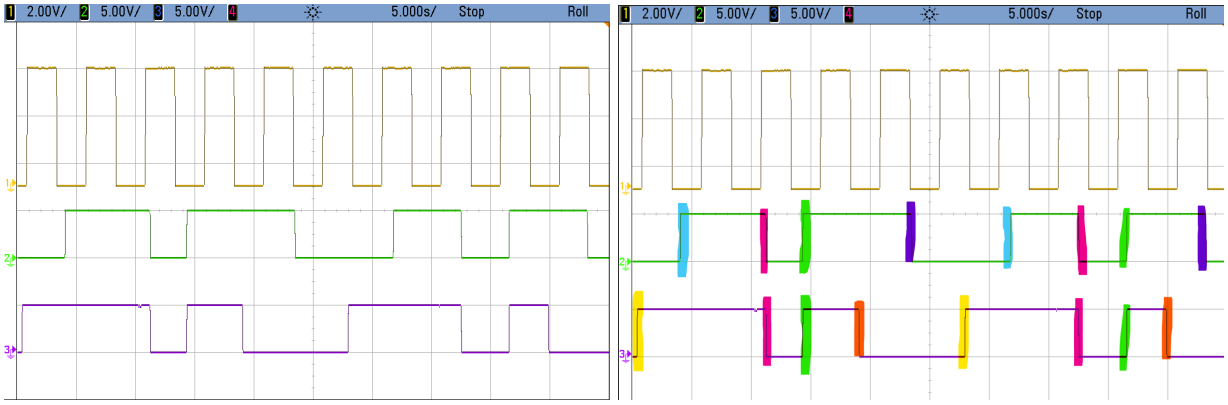

Figura 8: Señales de salida del circuito antes diferentes combinaciones de entradas.

Los colores que resaltan las transiciones indican lo siguiente:

- Amarillo: S=1 e I=1 → S=0 e I=1.
- Celeste: S=0 e I=1 → S=0 e I=0.
- Rojo: S=0 e I=0 → S=1 e I=0 (caso imposible).
- Verde: S=1 e I=0 → S=0 e I=0.
- Naranja: S=0 e I=0 → S=0 e I=1.
- Violeta: S=0 e I=1 → S=1 e I=1.

Puede apreciarse en esta secuencia el cambio de bomba que alternativamente le toca funcionar por sí sola.

Conclusiones

A modo de cierre de esta sección, puede afirmarse el correcto funcionamiento del sistema procedural mediante el cual, dado un problema que tomaba distintas acciones antes distintos "momentos.[en] su funcionamiento, se lo puede resolver mediante el planteo de una máquina de estados teórica y su posterior implementación mediante Flip-Flops y compuertas lógicas.
Ha de destacarse nuevamente la ventaja en rendimiento de Flip-Flops que supuso la resolución del problema mediante una máquina de Mealy por sobre una de Moore, adjudicando esta diferencia entre máquinas de estados a la alta dependencia de las salidas con las entradas.

Ejercicio 2: Reconocimiento de secuencia de bits

Se desea diseñar una máquina de estados implementada con máquina de Mealy, la cual sea capaz de analizar una secuencia de bits y detectar si se produjo un patrón seguido por 1-1-0-1, ante lo cual deberá notificar tal suceso activando su salida para ello. En la Fig. **??** se ilustra un esquema general de ello.

Diseño de Máquina de Estados

En primer lugar, dadas las especificaciones de la máquina de estado, se desea diseñar tal dispositivo el cual consta de una única entrada y una única salida. Para lo cual se emplea un esquema genérico de máquina de estados, en donde la salida será asincrónica pues se busca utilizar el diseño de Mealy para tal lógica. Este esquema general descripto puede visualizarse en la Fig. 9, donde la cantidad de entradas no es necesariamente la misma que en la salida.

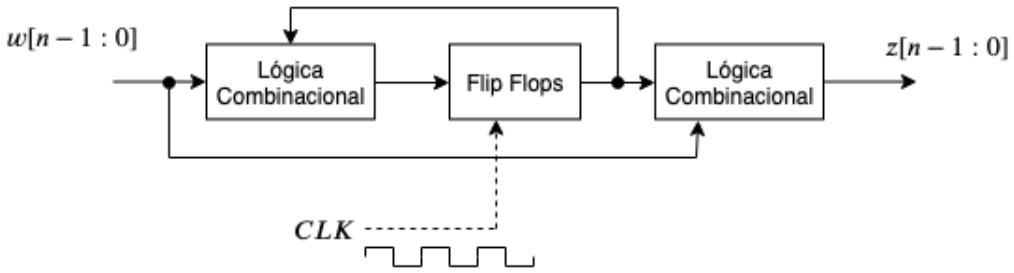

Figura 9: Esquema general de la máquina de Mealy

En la Fig. 10 se puede observar el diagrama de estados propuesto. Es importante mencionar que el estado de Reset es definido como tal para reconocer cuál es el estado inicial de la máquina, y deberá ser tenido en cuenta durante la asignación de estados en caso de proveer la posibilidad de reiniciar la máquina, pues los flip flops deberán ser llevados a dicho estado, según sea asignado.

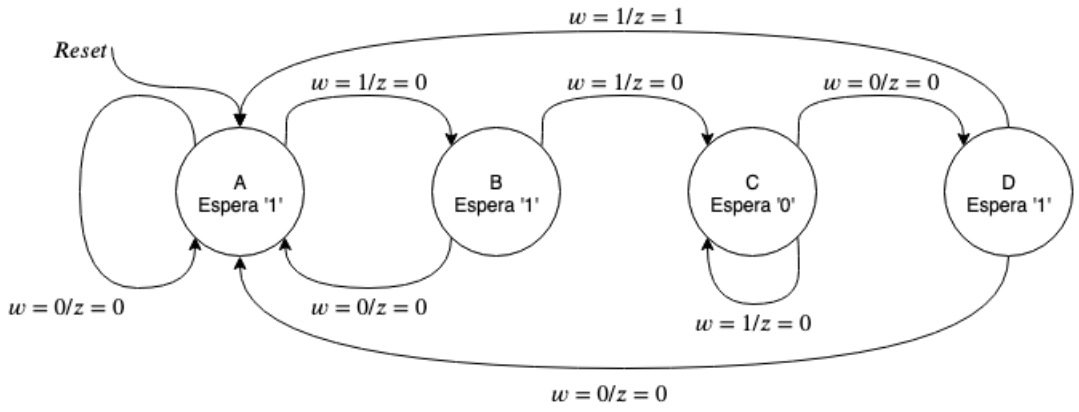

Figura 10: Diagrama de estados

En la Tabla 2 se puede observar la tabla de estados o transiciones de la máquina de estados, habiendo ya asignado correspondientemente a cada estado una configuración de bits. Es de interés mencionar que tal asignación es el resultado de comparar las diferentes alternativas y encontrar que, dada la distribución propuesta, la lógica externa es la mínima necesaria.

Estado	Próximo		Salida	
y_2y_1	$w=0$	$w=1$	$w=0$	$w=1$
$A=11$	$A=11$	$B=00$	$z=0$	$z=0$
$B=00$	$A=11$	$C=01$	$z=0$	$z=0$
$C=01$	$D=10$	$C=01$	$z=0$	$z=0$
$D=10$	$A=11$	$A=11$	$z=0$	$z=1$

Tabla 2: Tabla de estados o transiciones

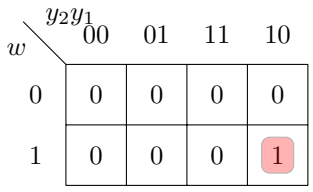

Figura 11: Karnaugh para la variable de salida

$$z = w \cdot y_2 \cdot \overline{y_1} \tag{1}$$

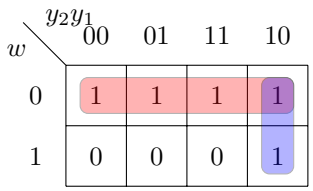

Figura 12: Karnaugh para la variable de estado y_2

$$y_2 = \overline{w} + y_2 \cdot \overline{y_1} \tag{2}$$

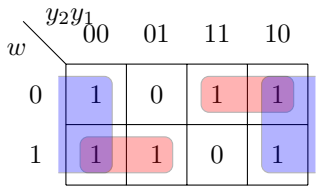

Figura 13: Karnaugh para la variable de estado y_1

$$y_1 = \overline{y_1} + \overline{w} \cdot y_2 + w \cdot \overline{y_2} \tag{3}$$

Simulaciones en Verilog

En la Fig. 14 se puede observar el circuito lógico completo correspondiente a la máquina de estados diseñada en el apartado anterior. De la misma forma que en el ejercicio 1, la simulación se realiza de dos

maneras: mediante un bloque procedural, y a través de la implementación de los circuitos lógicos que describen la máquina de estados.

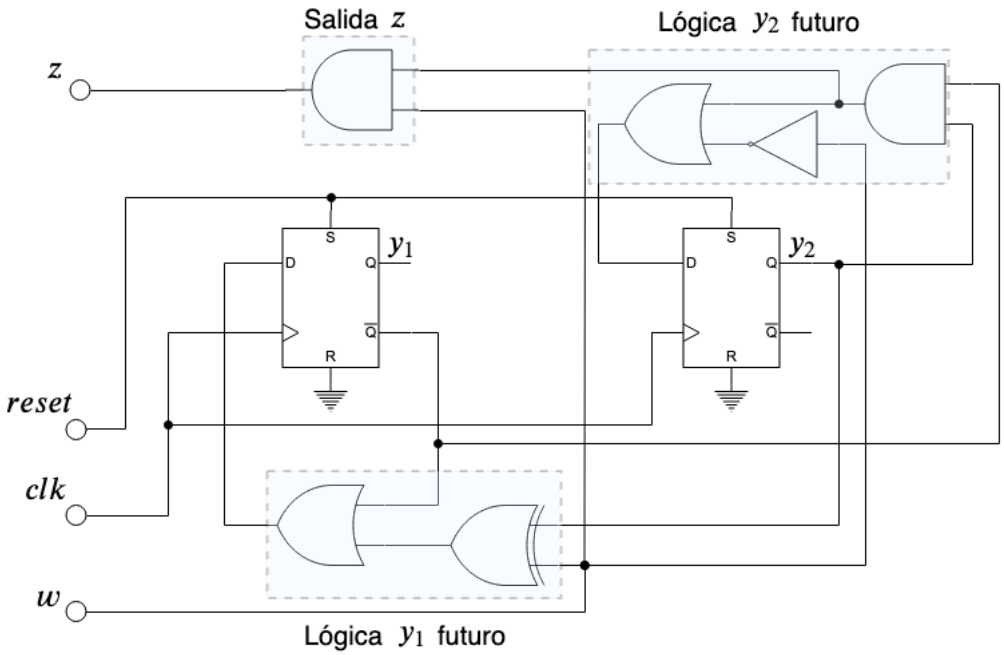

Figura 14: Circuito lógico completo de la máquina de estados

En la Fig. 15 se muestra a modo de referencia la simulación realizada y visualizada con GTKWave. Para determinar los casos de pruebas de la máquina, se partió del diagrama de estados y se consideraron diferentes secuencias. En primer lugar la secuencia 0-1-0 para determinar si permanece correctamente en el primer estado, pasa al segundo y vuelve al detectar el error. En segundo lugar, la secuencia 1-1-1-0-0, para determinar si llega correctamente al tercer estado, permanece y luego transiciona reiniciando la máquina pero con la salida en estado bajo. Finalmente, la secuencia correcta para analizar si la salida responde como es esperado.

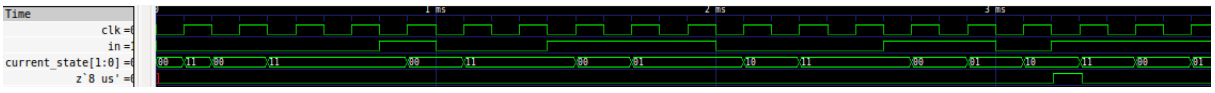

Figura 15: Simulación de Verilog visualizada con GTKWave

Diseño en PCB

La realización del PCB para implementar el circuito lógico implica la conexión de las compuertas lógicas siguiendo el esquema teórico, y verificando previamente que no hayan complicaciones físicas en tales conexiones. Esto último implica utilizar consistentemente tecnología TTL, verificando que las corrientes de salida de las compuertas no sea superada con el consumo de las entradas, es decir no superar el fan-out. Además, para prevenir fallos por picos en la tensión de alimentación durante transitorios de las compuertas, se conectaron los debidos capacitores de desacople.

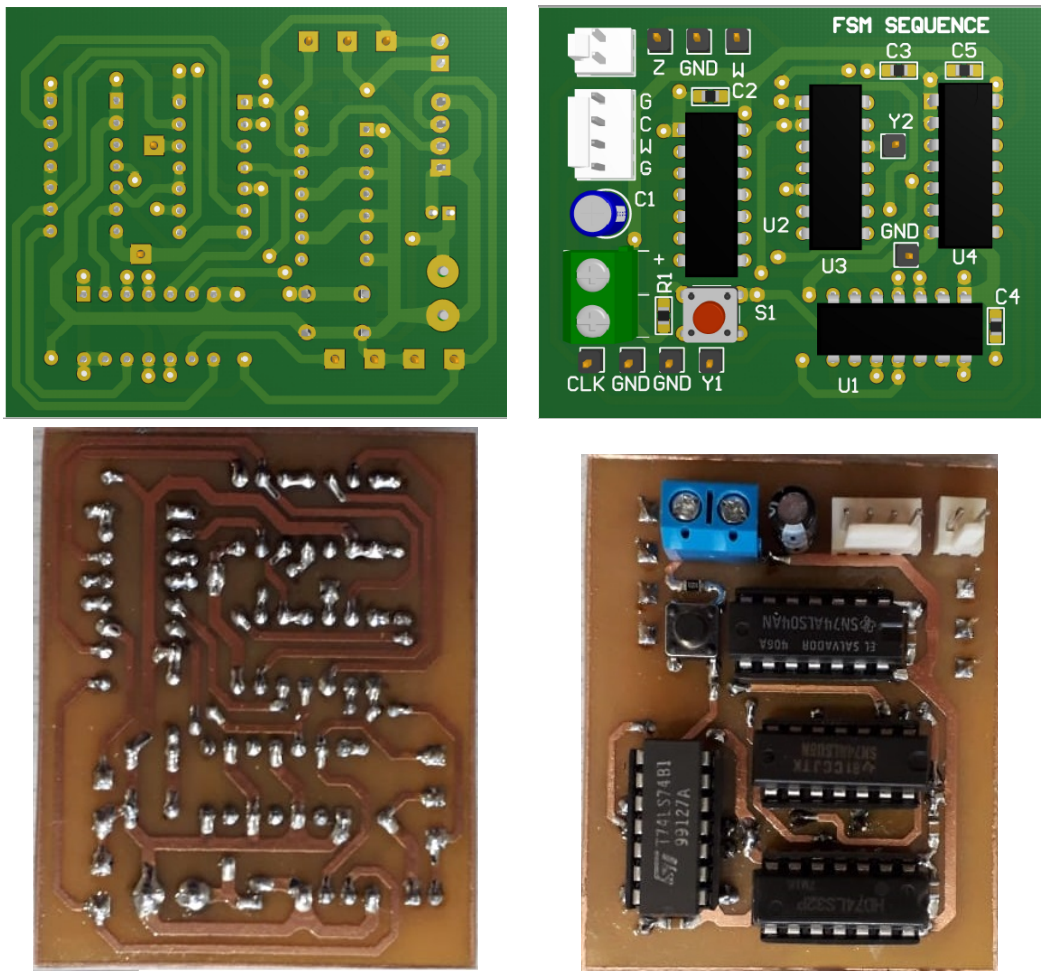

Figura 16: PCB diseñado e implementado

Resultados

Para la verificación del correcto funcionamiento se propone poner bajo prueba al circuito con las mismas tres secuencias que fueron empleadas en el proceso de simulación. Estas secuencias son 0-1-0, 1-1-1-0-0 y 1-1-0-1, para lo cual se utiliza una señal de clock de baja frecuencia y la entrada se controla con un pulsador externo al PCB, y se realizan estas mediciones en dos partes para poder extraer la información de la salida y la de los estados, ya que sólo se dispone de osciloscopio de cuatro canales.
En la Fig. 17 se muestran las mediciones de los estados de la máquina de estados, en donde las señales Amarilla, Verde, Azul y Roja/Rosa, corresponden respectivamente a la señal de clock, la entrada w, el bit de estado y_2 y el bit de estado y_1. Mientras que en la Fig. 18 se mide la salida de la máquina de estados, en donde la señales Amarilla, Verde y Azul, corresponden respectivamente a la señal de clock, la entrada w y la salida z.
De izquierda a derecha, de arriba hacia abajo se ordenan los casos de cada secuencia según fueron mencionados anteriomente.

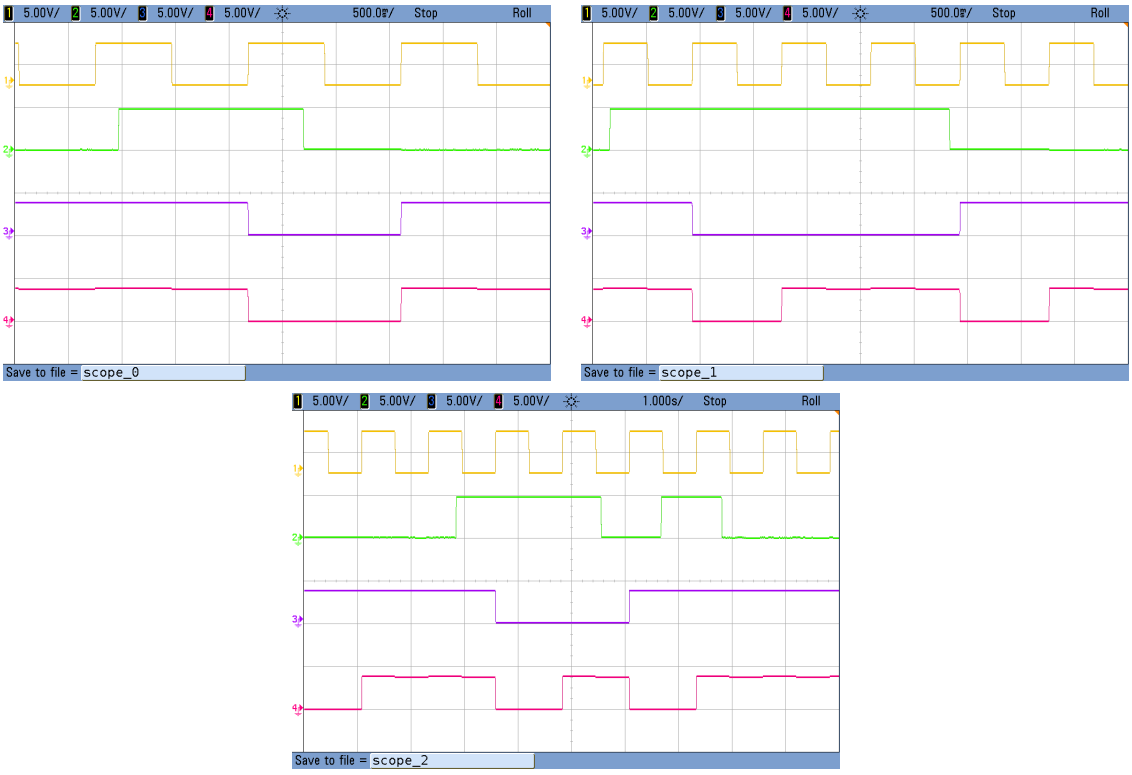

Figura 17: Estados de la FSM

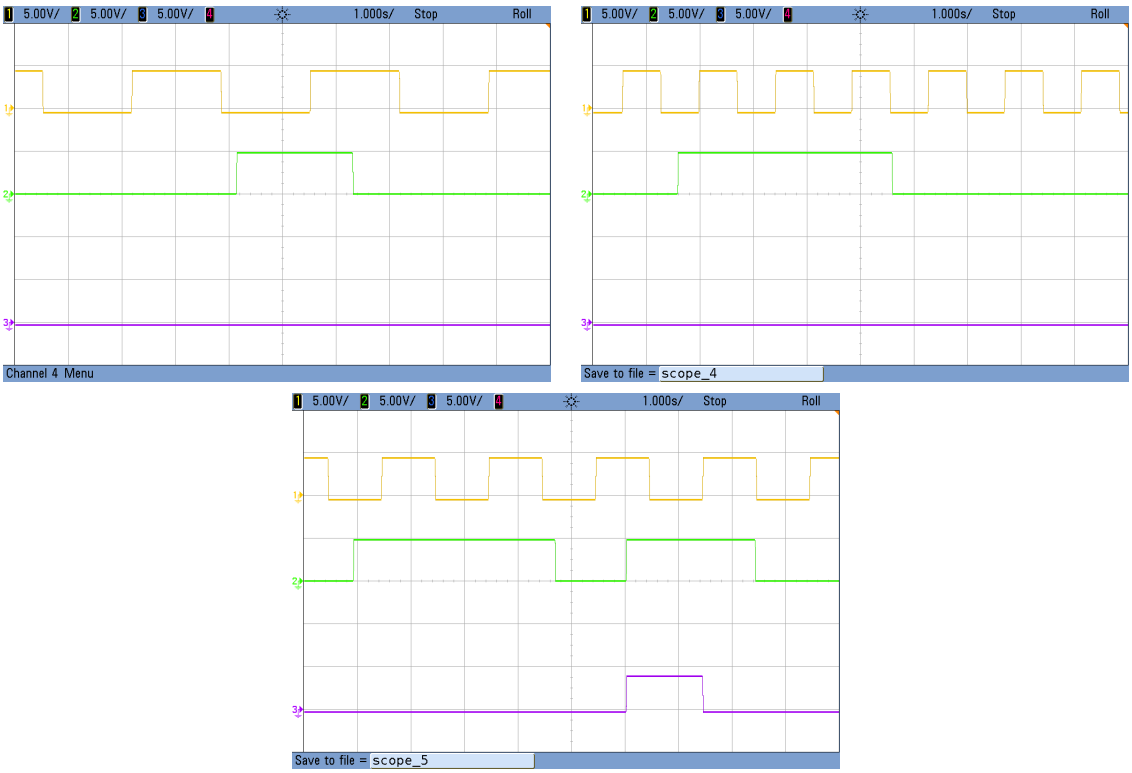

Figura 18: Salida de la FSM

Conclusiones

Considérese la máquina de estados diseñada en base al reconocimiento de una secuencia de 4 bits, luego para el caso de una máquina de Moore se necesitaría un estado adicional donde luego de detectar la secuencia se mantenga la salida en un estado alto. Por otro lado, considerando los tiempos de propagación, para el instante del flanco ascendente en el cual la secuencia correcta es identificada, la salida no necesariamente refleja tal detección sino hasta pasar un determinado tiempo, como consecuencia de su sincronismo. Estos aspectos evidencian por qué una máquina de Mealy puede resultar beneficiosa, dado que hace uso de menos estados, y su salida al ser asincrónica permite que para el flanco de detección de la secuencia la salida refleje haber detectado correctamente el patrón.

Ejercicio 3: Máquina de Moore

En esta sección se desea diseãr una máquina de estados implementada con mąquina de Moore que cumpla con la mostrada en la Figura 19.

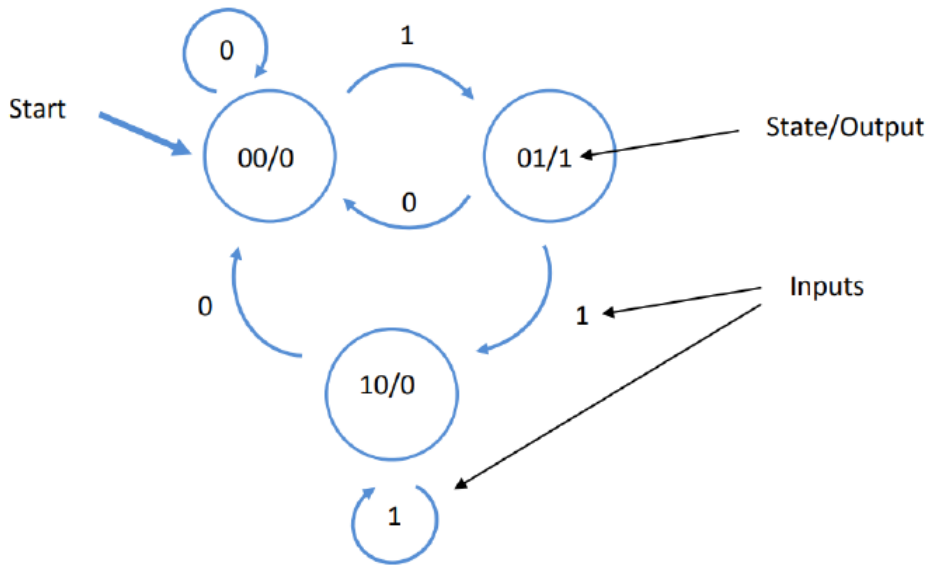

Figura 19: Máquina de estados en base a la cual se realiza el diseño

Este comportamiento se corresponde con un detector de flancos que mantiene su salida en alto por un ciclo de *clock*.

Diseño de Máquina de Estados

Se puede observar en la Tabla 3 la tabla de estados correspondiente a la máquina presentada en la Figura 19. Cabe aclarar que, en esta implementación se decide no fijar la salida en los estados no utilizados y utilizarlos como *don't care* para facilitar el diseño.

Estado Actual	Estado Siguiente		Salida
y_2y_1	$\omega = 0$ Y_2Y_1	$\omega = 1$ Y_2Y_1	Z
00	00	01	0
01	00	10	1
10	00	10	0
11	xx	xx	x

Tabla 3: Tabla de estados de la máquina de Moore

Al haber 3 estados, es necesario utilizar 2 Flip Flops, en este caso D, para implementarla. Se resuelven entonces los mapas de Karnaugh para obtener el circuito que resuelve la Tabla 3.

w \ y_2y_1	00	01	11	10
0	0	0	X	0
1	1	0	X	0

Figura 20: Karnaugh para el estado siguiente Y_1

De este mapa se obtiene la solución que se muestra en 4

$$Y_1 = \omega \cdot \overline{y_1} \cdot \overline{y_2} \tag{4}$$

w \ y_2y_1	00	01	11	10
0	0	0	X	0
1	0	1	X	1

Figura 21: Karnaugh para la variable Y_2

De este mapa se obtiene la solución que se muestra en 5

$$Y_2 = \omega \cdot y_1 + \omega \cdot y_2 \tag{5}$$

Además, es posible observar en la tabla de estados que $Z = y_1$. Si bien esto fija el valor de salida en el caso de que el estado actual fuera $y_2 - y_1 = 1 - 1$, como en principio este estado no es válido nunca debería suceder. Mas allá de esto, se define la salida como *don't care* en ese caso, así que tampoco presenta un problema. Se presenta entonces, en la Figura **??**, el circuito lógico obtenido a partir de resolución anterior.

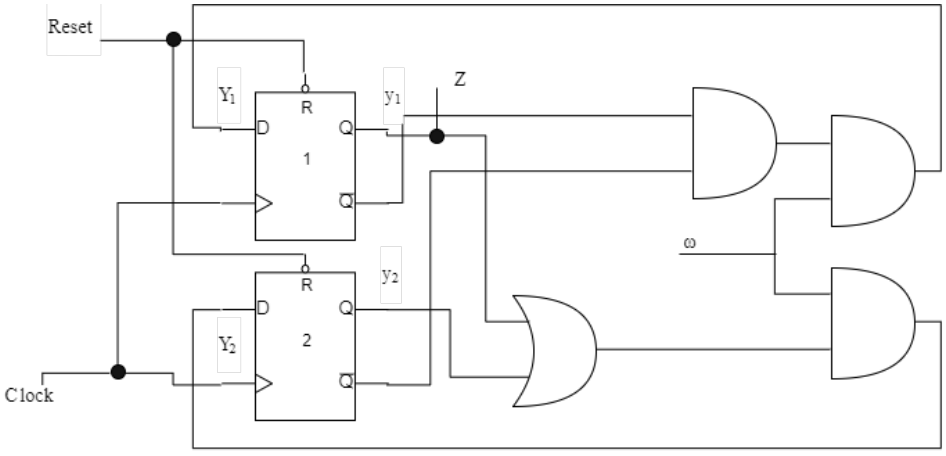

Figura 22: Circuito implementado

Se puede observar que se agrega una entrada adicional de reset, para poder resetear los flipflops al iniciar, comenzando en un estado válido

Level Shifters

Si bien entradas y salidas del circuito son de 5V-0V, se diseña la lógica interna de la máquina de estados para que funcione a 3.3V. Se utilizan con este fin *level shifters* en las entradas y salidas del sistema. Luego de contrastar el funcionamiento de las opciones disponibles que cumplen con lo requerido, se concluye que la de mejor funcionamiento en cuanto a sus características, como tiempo de propagación y tiempo de rise, es presentada en la Figura 23.

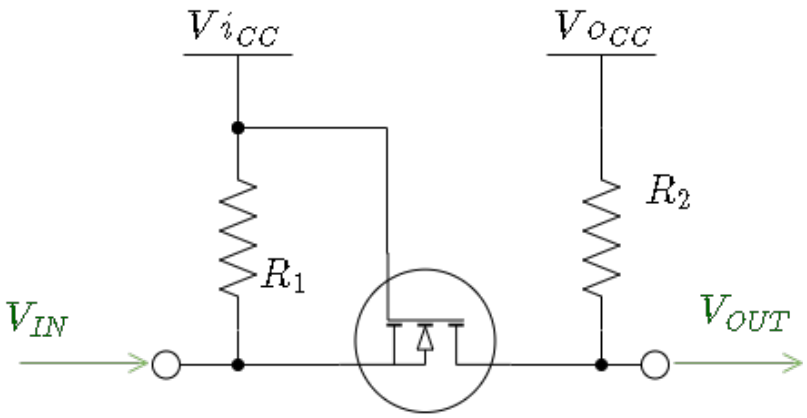

Figura 23: Level shifter utilizado

Regulador de tensión

Para obtener una tensión constante en 3.3V, que se utilizan tanto para la alimentación de los circuitos integrados, como para los *level shifters*, se implementa un regulador de tensión simple con un diodo zener y una resistencia. A pesar de que el regulador funciona correctamente, se observa como un fallo en su diseño que la corriente que consume es muy elevada. Se asume este defecto a la baja resistencia utilizada en el regulador.

Mediciones y resultados

Se puede ver en la Figura 24 se puede ver como responde la máquina de estados ante una entrada que se mantiene en alto por menos de un ciclo de clock, es decir, del estado 0-0 al estado 0-1 y de nuevo al estado 0-0.

Figura 24: del estado 0-0 al estado 0-1 y de nuevo al estado 0-0..Entrada en azul, clock en rojo y salida en verde

En la Figura 25 se observa la respuesta de la máquina de estados a una entrada que se mantiene en alto por más tiempo del que dura un período del clock, es decir la transición 0-0, 0-1, 1-0.

Figura 25: Transición 0-0, 0-1, 1-0. Entrada en azul, clock en rojo y salida en verde

Se puede observar que en todos los casos el sistema responde en concordancia con lo esperado.