Laboratorio 4: Lógica Secuencial y Controladores

Fiorella Delgado León, Jonathan Guzmán Araya, Gerald Valverde Mc kenzie
fiorelladelgado53@gmail.com, jonathana1196@gmail.com, gvmckenzie@mckode.com
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica, Cartago, Costa Rica, 2021

I. Introducción

Un circuito secuencial es aquel en el cual las salidas en un instante de tiempo determinado son una función de las entradas en ese instante y en instantes anteriores, por lo que se dice que este tipo de circuitos es capaz de "memorizar" información.

Una máquina de estados finitos es un modelo matemático de computación en un momento dado del tiempo, se encuentra en un estado, y realiza cómputos de forma automática sobre una entrada para producir una salida. Este modelo está conformado por un alfabeto Σ, un conjunto de estados finitos Q, una función de transición δ, un estado inicial $q_i \in Q$, una posible función de salida Ω, y un conjunto de estados finales A. Su funcionamiento se basa en su función de transición, que recibe a partir de un estado inicial y que va avanzando en el autómata de un estado a otro causando una transición entre los estados dependiendo de la entrada que recibe, para finalmente detenerse en un estado final. La función de transferencia de una máquina de estados puede ser parcial, es decir, puede no estar definida para toda combinación de $q \in Q$ y $\sigma \in \Sigma$.

El origen de los autómatas finitos probablemente se remonta a su uso implícito en máquinas electromecánicas, desde principios del siglo XX wolfram. Ya en 1907, el matemático ruso Andréi Márkov formalizó un proceso llamado cadena de Markov, donde la ocurrencia de cada evento depende con una cierta probabilidad del evento anterior basharin. Esta capacidad de "recordar" es utilizada posteriormente por los autómatas finitos, que poseen una memoria primitiva similar, en que la activación de un estado también depende del estado anterior, así como del símbolo o palabra presente en la función de transición. Posteriormente, en 1943, surge una primera aproximación formal de los autómatas finitos con el modelo neuronal de McCulloch-Pitts. Durante la década de 1950 prolifera su estudio, frecuentemente llamándoles máquinas de secuencia; se establecen muchas de sus propiedades básicas, incluyendo su interpretación como lenguajes regulares y su equivalencia con las expresiones regulares wolfram. Al final de esta década, en 1959, surge el concepto de autómata finito no determinista en manos de los informáticos teóricos Michael O. Rabin y Dana Scott. En la década de 1960 se establece su conexión con las series de potencias y los sistemas de sobre-escritura. Finalmente, con el desarrollo del sistema operativo Unix en la década de 1970, los autómatas finitos encuentran su nicho en el uso masivo de expresiones regulares para fines prácticos, específicamente en el diseño de analizadores léxicos, la búsqueda y reemplazo de texto.

Se pueden clasificar en:

- Deterministas
  - La transición desde un estado puede tener como destino un único estado.
  - No se aceptan transiciones con cadenas vacías.
  - Se permite el uso de backtracking.
  - Una cadena es aceptada si su transición es hacia un estado final.
- No deterministas
  - La transición desde un estado puede tener múltiples destinos.
  - Permite transiciones con cadenas vacías.
  - Una cadena es aceptada si solo una de todas sus posibles transiciones son hacia un estado final.

Adicionalmente, las máquinas de estado pueden ser clasificadas como Máquinas de Mealy, cuya salida depende de su estado actual y de la entrada, es decir $\Omega : Q \times \Sigma \to \Lambda$ ó Máquinas de Moore, cuya salida depende solamente de su estado actual, es decir $\Omega : Q \to \Lambda$.

Los autómatas finitos se pueden representar mediante grafos particulares, también llamados diagramas de estados finitos, de la siguiente manera:

- Los estados *Q* se representan como vértices, etiquetados con su nombre interior.
- Una transición δ desde un estado a otro, dependiente de un símbolo del alfabeto, se representa mediante una arista dirigida que une a estos vértices, y que no está etiquetada con dicho símbolo.
- El estado inicial q_0 se caracteriza por tener una arista que llega a él, proveniente de ningún otro vértice.
- El o los estados finales *F* se representan mediante vértices que están encerrados a su vez por otra circunferencia.

El autómata finito en la Figura 1 está definido sobre el alfabeto $\Sigma = \{0, 1\}$, posee dos estados, S_1 y S_2, y sus transiciones son $\delta(S_1, 0) = S_2$, $\delta(S_1, 1) = S_1$, $\delta(S_2, 0) = S_1$, y $\delta(S_1, 1) = S_2$. Su estado inicial es S_1, que también es su único estado final.

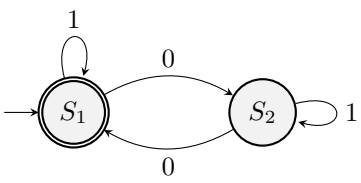

Fig. 1. Autómata que determina si un número binario tiene un cantidad par de ceros

Otra manera de describir el funcionamiento de un autómata finito es mediante el uso de tablas de transiciones o matrices de estados. Para el ejemplo de la Figura 1 la Tabla I y la Tabla II son posibles representaciones alternativas.

Salida $q \in Q$	Símbolo $\sigma \in \Sigma$	Llegada $\delta(q, \sigma) \in Q$
S_1	0	S_2
S_1	1	S_1
S_2	0	S_1
S_2	1	S_2

TABLE I
TABLA DE TRANSICIONES

	0	1
$\rightarrow\star S_1$	S_2	S_1
S_2	S_1	S_2

TABLE II
TABLA DE TRANSICIONES

En la Tabla II se marca el estado inicial con $\rightarrow$ y el estado final con $\star$, mientras que en la tabla se muestra el estado actual y la entrada que causa la transición al siguiente estado.

La Figura 2 muestra una máquina de Moore; un tipo de máquina de estados finitos que genera una salida basándose en su estado actual, para cada estado S, la salida aparece en el nodo, por ejemplo, para el estado *A*, la salida 0 aparece como A/0. Para este ejemplo, se muestra una red secuencial, que tiene una entrada, y una salida. La salida es 1, y se mantiene en 1, cuando se han introducido al menos dos 0s, y dos 1s como entrada.

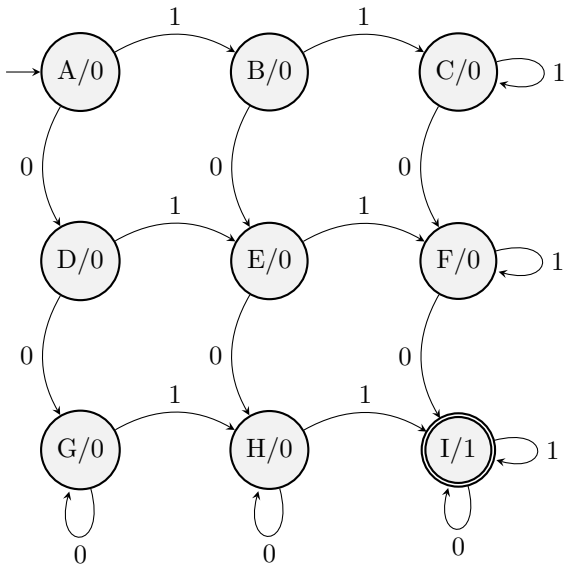

Fig. 2. Diagrama de estados de una máquina de Moore

Como se puede ver en la tabla III, el valor de la salida sólamente depende del estado actual.

En contraste, como se puede ver en la tabla IV, el valor de la salida depende tanto del estado actual, como de la entrada, a excepción de la transición del estado inicial S_i, pero eso se debe al ejemplo particular.

Estado actual	entrada	Siguiente estado	Salida
A	0	D	0
	1	B	
B	0	E	0
	1	C	
C	0	F	0
	1	C	
D	0	G	0
	1	E	
E	0	H	0
	1	F	
F	0	I	0
	1	F	
G	0	G	0
	1	H	
H	0	H	0
	1	I	
I	0	I	1
	1	I	

TABLE III
TABLA DE TRANSICIÓN DE RED SECUENCIAL, CUYA SALIDA ES 1 CUANDO HA TENIDO DOS 0S Y DOS 1S DE ENTRADA

La figura 3 muestra una máquina de Mealy; un tipo de máquina de estados finitos que genera una salida basándose en su estado actual y una entrada. Su diagrama de estados incluye ambas señales de entrada (en rojo) y de salida (en verde) para cada línea de transición. Esta empieza en el estado S_i, e implementa la función *XOR* de los últimos dos valores más recientemente introducidos.

En contraste, la salida de una máquina de Moore de estados finitos (el otro tipo) depende solamente del estado actual de la máquina, dado que las transiciones no tienen entrada asociada. Sin embargo, para cada máquina de Mealy hay una máquina de Moore equivalente cuyos estados son la unión de los estados de la máquina de Mealy y el producto cartesiano de los estados de la máquina de Mealy y el alfabeto de la entrada.

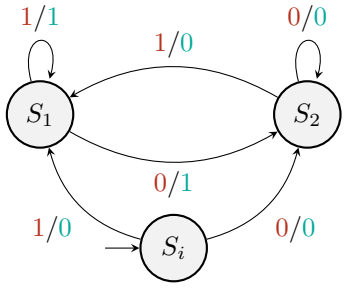

Fig. 3. Diagrama de estados para una Máquina de Mealy simple, con una entrada, y una salida

Los verificadores de tiempo se utilizan para verificar que en cierto intervalo de tiempo, la señal se debe mantener estable, antes y después de que la entrada del clock cambie.

Estado actual	entrada	Siguiente estado	Salida
S_i	0	S_0	0
	1	S_1	
S_0	0	S_0	0
	1	S_1	1
S_1	0	S_0	1
	1	S_0	0

TABLE IV
TABLA DE TRANSICIÓN DE UN *XOR* DE LAS ÚLTIMAS 2 ENTRADAS

- Setup time: Es la cantidad de tiempo en el que la señal sincronizada debe estar estable antes que el clock tenga un cambio, para garantizar que los datos se guarden/transmitan exitosamente, este valor puede ser modificado, variando el periodo en el clock.
- Hold time: Es la cantidad de tiempo en que la señal se mantiene estable después de haber tenido el cambio y haber pasado el borde del clock. Este también garantiza que los datos se guarden/transmitan exitosamente.

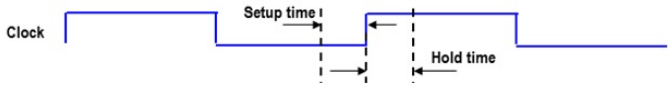

Fig. 4. Setup Time & Hold Time

II. DESARROLLO

Para el desarrollo de este laboratorio, se realizaron tres experimentos, en los cuales se pone en práctica la implementación de los lenguajes System Verilog y VHDL Verilog.

A. Experimento 1

En este experimento, se requiere realizar ...

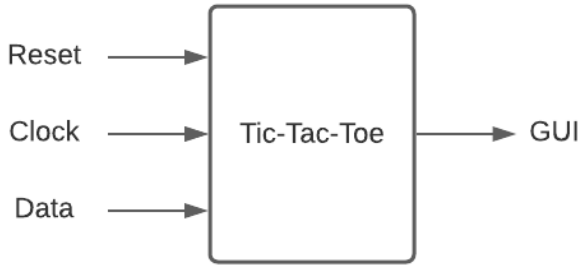

Fig. 5. Diagrama de primer nivel de Tic-Tac-Toe.

III. RESULTADOS

A. Experimento 1

IV. ANÁLISIS DE RESULTADOS

A. Experimento 1

Como se puede observar en la Figura

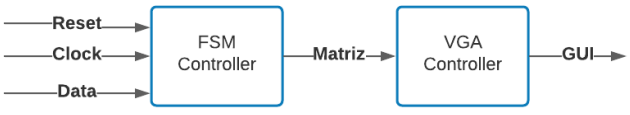

Fig. 6. Diagrama de segundo nivel de Tic-Tac-Toe.

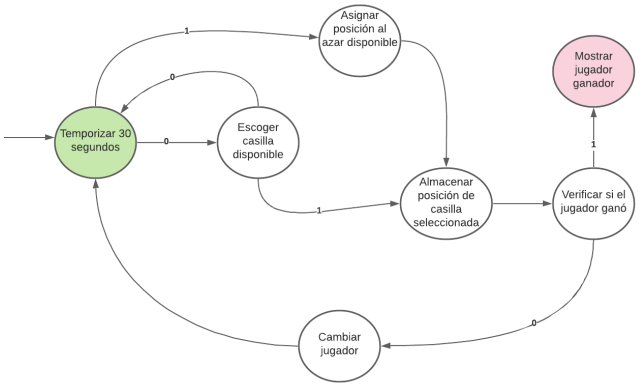

Fig. 7. Maquina de estados del programa.

V. CONCLUSIONES

Al desarrollar este laboratorio, se obtuvieron las siguientes conclusiones:

- Al implementar cada uno de los tres experimentos, se puso en práctica el modelo de comportamiento y estructural.
-