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

***Resumen*—In this laboratory it´s put into practice the work with FPGAs, their respective configuration in the System Verilog and VHDL language also an introduction to VGA controllers was made, as well as the concept of finite state machines was remembered and used both in its behavioral and structural form in VHDL for problem solving.**

***Palabras clave*—Backtracking, clock, comportamiento, controladores, estructural, Flip-Flop, FPGA, FSM, reset, RGB, síntesis lógica y testbench. VGA.**

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 [3]. 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 [**?**] . 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 [3]. 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 $\Omega : Q \times \Sigma \rightarrow \Lambda$ ó Máquinas de Moore, cuya salida depende solamente de su estado actual, es decir $\Omega : Q \rightarrow \Lambda$.

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.

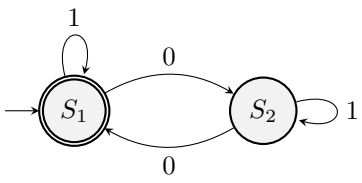

Figura 1. Autómata que determina si un número binario tiene un cantidad par de ceros

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

Cuadro I
TABLA DE TRANSICIONES

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

Cuadro 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.

Como se puede ver en la Tabla III, el valor de la salida solamente 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.

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

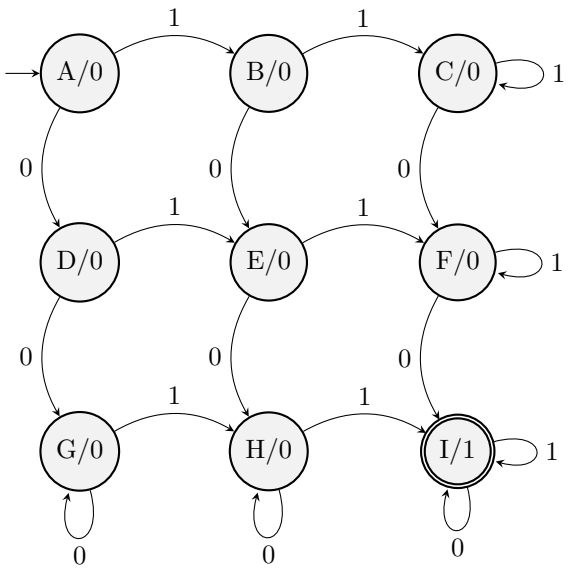

Figura 2. Diagrama de estados de una máquina de Moore

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	

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

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.

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.

- Setup time: Es la cantidad de tiempo en el que la señal sincronizada debe estar estable antes que el clock

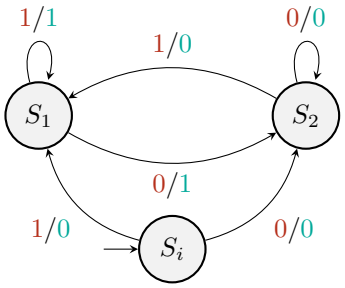

Figura 3. Diagrama de estados para una Máquina de Mealy simple, con una entrada, y una salida

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

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

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.

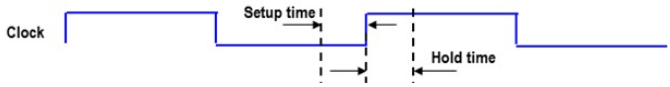

Figura 4. Setup Time & Hold Time [5]

Los elementos mecánicos como botones, teclas o interruptores, son dispositivos que cierran o abren un circuito mediante el contacto entre dos superficies metálicas, tal y como se muestra en la siguiente figura.

Figura 5. Dispositivos mecánicos [5]

Dichas superficies metálicas, poseen elasticidad, de manera que al ponerlas en contacto se genera un choque que produce un movimiento en sentido contrario que las aleja, lo cual, sucede repetidamente hasta que se disipa la energía cinética adquirida por la lámina en movimiento, a este fenómeno se le conoce como efecto rebote. El efecto rebote se presenta por el hecho de que la apertura o el cierre del circuito no es instantáneo, por lo que durante un intervalo de tiempo oscila entre cerrado y abierto.

En el caso de los circuitos digitales, también existen soluciones para este problema, por ejemplo, se puede hacer un circuito

Figura 6. Dispositivos mecánicos [5]

anti-rebote, de manera que se utilizan componentes como un Flip-Flop tipo R-S, de manera que, si se tienen ambas entradas conectadas a la tierra, con resistencias, se puede cambiar su salida como respuesta a un cambio en una de sus entradas, pero como los flips-flops tienen el efecto de memoria, no importa, porque su estado se mantiene igual a que si solo hubiera entrada un pulso de 1 a 0, hasta que se haga más estable, también se usan compuertas lógicas , especialmente las de ”smith-trigger”, ya que son mucho mas estables.

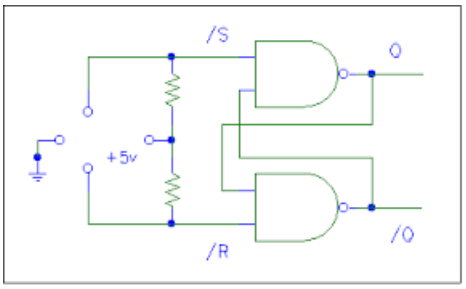

Figura 7. Ejemplo Solución Flip-Flop [5]

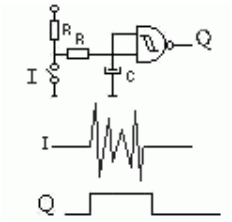

Figura 8. Ejemplo solución con smith-trigger [5]

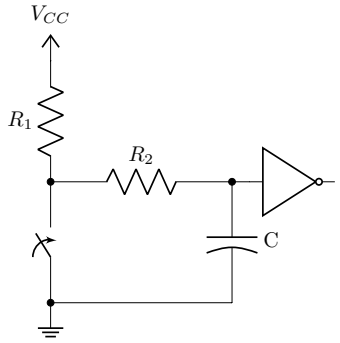

Figura 9. Circuito RC para remover oscilación de señal

Las señales de sincronización de una interfaz VGA son las que se detallan en la tabla 1. R, G y B son tres señales analógicas que que determinan el color de un punto en la

pantalla. Por otra parte, h_sync y v_sync son las señales que determinan la posición de referencia de la pantalla donde debe ser mostrado el punto.

Pin	Señal	Descripción
1	R	análogo rojo, 0-0.7 V
2	G	análogo verde, 0-0.7 V o 0-3.1 V si sync-on-green
3	B	análogo azul, 0-0.7 V
13	h_{sync}	horizontal sync, 0V/5V waveform
14	v_{sync}	vertical sync, 0V/5V waveform

Cuadro V
TABLA DE TRANSICIONES

Mediante un manejo correcto de estas señales según las especificaciones de sincronización de la VGA, es posible mostrar lo que se desee en la pantalla.

Un diagrama de tiempos de las señales de sincronización de VGA para una resolución de 640x480 píxeles esta dado de la forma:

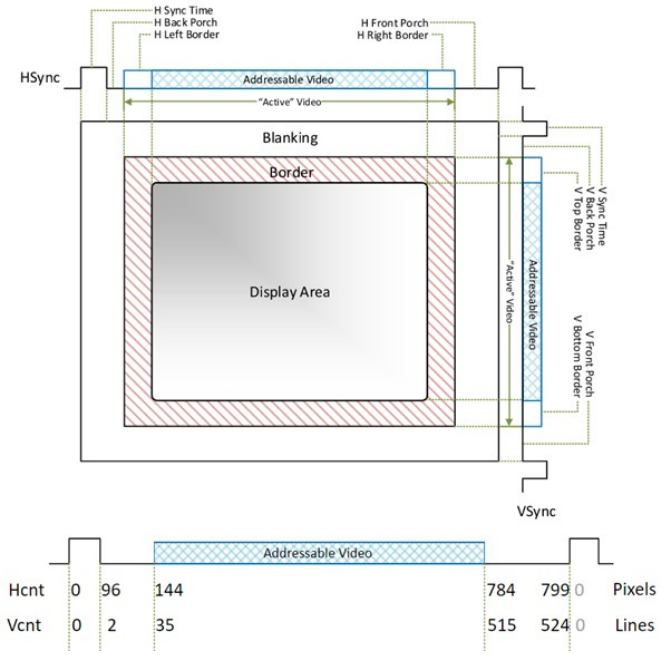

Figura 10. VGA [6]

De esta manera para la sincronización horizontal se tiene:

- $HSyncTime = 3{,}813\mu s$
- $HBackPorch = 1{,}907\mu s$
- $HFrontPorch = 0{,}636\mu s$
- $HAddrVideoTime = 25{,}422\mu s$
- $HL/RBorder = 0\mu s$
- $Total = 31{,}778\mu s$

Realizando el calculo de la frecuencia tenemos:

$$f_H = \frac{1}{31{,}778\mu s} = 31{,}468kHz \quad (1)$$

Y para la sincronización vertical se tiene:

- $VSyncTime = 0{,}064ms$
- $VBackPorch = 1{,}048ms$
- $VFrontPorch = 0{,}318ms$
- $VAddrVideoTime = 15{,}253ms$
- $VT/BBorder = 0ms$
- $Total = 16{,}683ms$

Realizando el calculo de la frecuencia tenemos:

$$f_H = \frac{1}{16{,}683ms} = 59{,}94Hz \quad (2)$$

- Front Porch: Es una región en blanco de la señales de sincronización horizontal y vertical, que se presenta antes de la región activa de vídeo.
- Back Porch: Es una región en blanco de la señales de sincronización horizontal y vertical, que se presenta después de la región activa de vídeo.

II. Sistema Desarrollado

II-A. *Experimento 1*

En este experimento, se requiere realizar ...

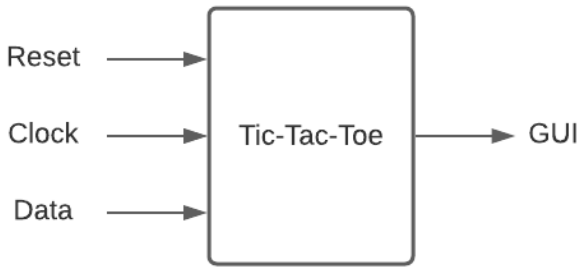

Figura 11. Diagrama de primer nivel de Tic-Tac-Toe.

Figura 12. Diagrama de segundo nivel de Tic-Tac-Toe.

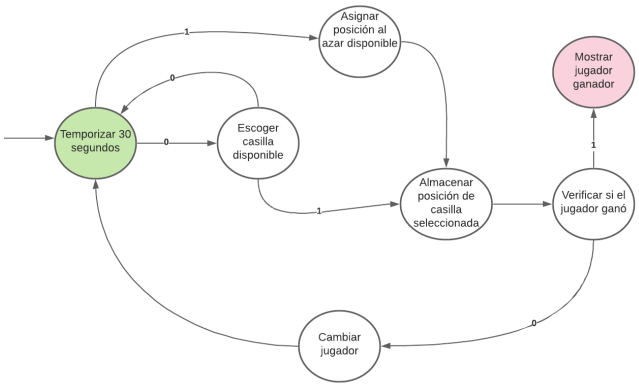

Figura 13. Maquina de estados del programa.

En la figura 13, se puede observar la máquina de estados del sistema, se requieren 3 flip flops a lo interno para poder

implementar una máquina como estas ya que al componerse de 7 estados, con 3 flip flops alcanza para poder trabajar con ella. El estado 000 es el Temporizador de 30 segundos, este es el estado inicial, ya que en cada una una vez que empieza la jugada, se deben empezar a contabilizar estos 30 segundos. El estado 001 es Escoger casilla disponible, la función de este estado es permitirle al jugador escoger una casilla disponible en la cuadrícula, si aún no pasan los 30 segundos. El estado 010 es Asignar posición al azar disponible, la función de este estado es la de asignar a un jugador una posición en la cuadrícula del juego disponible, si este no seleccionó ninguna, pasados los 30 segundos. El estado 011 es Almacenar posición de casilla seleccionada, su función es actualizar el vector de las posiciones de las casillas, con la que fue seleccionada, ya sea por el jugador o de manera aleatoria por el sistema. El estado 100 es Verificar si el jugador ganó, su función es verificar entre las jugadas permitidas ganadoras, y ver si el jugador con la selección de casilla anterior, generó alguna jugada ganadora. El estado 101 es Cambiar jugador, su función es asignarle el turno al siguiente jugador, en caso de que el que estaba jugando la partida, no haya generado una partida ganadora. Y finalmente, el estado 110 es Mostrar jugador ganador, este estado es el encargado de enviarle una señal al controlador de VGA, para indicarle que el jugador que está realizando dicha jugada, fue el ganador, ya que cumplió con alguna de las formas para poder ganar el juego (línea horizontal, línea vertical o diagonal).

III. Resultados

Al realizar este laboratorio, se obtuvo la máquina de estados que se observa en la Figura 14, generada al visualizar el diagrama de bloques de todo el sistema en RTL Simulation.

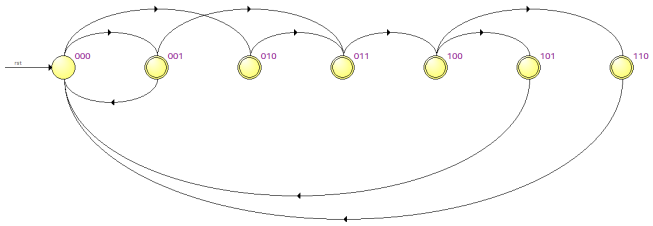

Figura 14. Máquina de estados del programa generada por RTL Simulation.

En esta máquina se puede observar como se cumple la lógica del juego descrita en la Figura 13.

Posteriormente, en la Figura 15, se puede observar la máquina de estados encargada de cambiar de jugador, esta se debió implementar para tener una lógica secuencial en el proceso de asignar el turno a cada jugador, e intercalarlos.

En la Figura 16, se puede observar el testbench generado al comprobar el funcionamiento del temporizador, este módulo es el encargado de contar los 30 segundos, sincronizado a la señal de reloj de todo el sistema.

En la Figura 17, se puede observar el testbench generado para comprobar el adecuado funcionamiento del contador de 1 a 9. Este módulo es el que es utilizado para asignar a la casilla correspondiente, su respectivo valor.

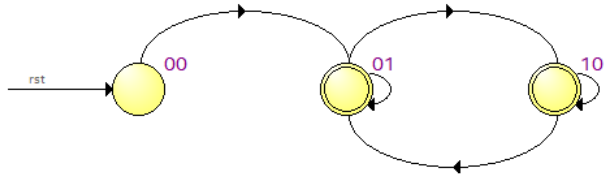

Figura 15. Máquina de estados para cambiar de jugador, generada por RTL Simulation.

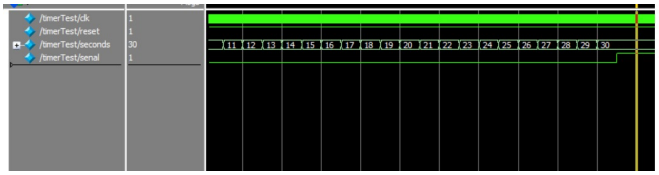

Figura 16. Testbench del temporizador.

En la Figura 18, se puede observar el testbench del módulo Random, el cual se encarga de asignar una posición de casilla random disponible, en caso de que pasen los 30 segundos y el jugador no haya seleccionado una casilla disponible.

En la Figura 19, se puede observar el testbench de la máquina de estados de los jugadores, en donde el estado 01 es para el jugador 1 y el estado 10, para el jugador 2; dicha máquina va a estar operando entre ambos estados, y cuando un jugador selecciona una casilla disponible, le da el turno al siguiente jugador para que escoja su respectiva casilla.

IV. Análisis de Resultados

Al implementar la lógica de la máquina de estados con el controlador VGA, existieron dos módulos los cuales no pudieron ser implementados con toda la lógica del programa: el timer, el cual se observa en la Figura 16 y el cual funciona adecuadamente por separado y con su respectivo testbench se puede ver su adecuado funcionamiento. Y por consiguiente el random, ya que no existe la señal que se genera por el temporizador al llegar a los 30 segundos para poner en alto su salida y así activar el módulo random, el cual se encarga

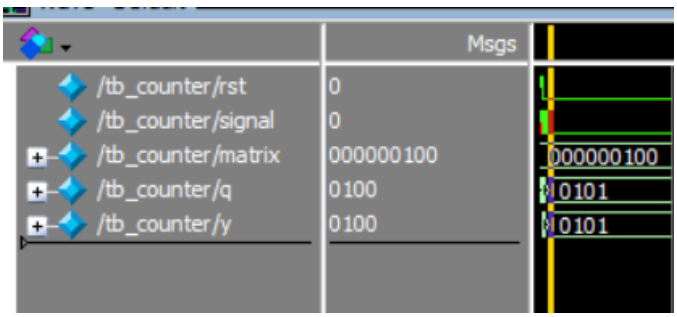

Figura 17. Testbench del contador.

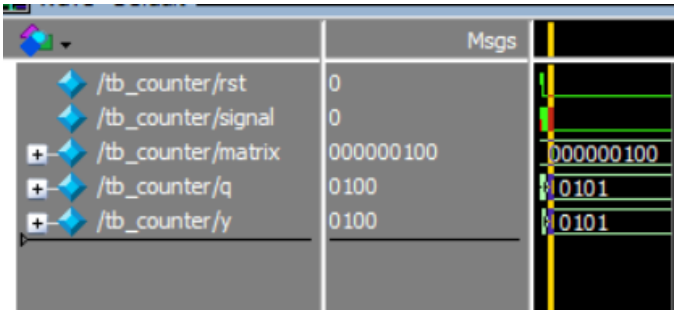

Figura 18. Testbench del Random.

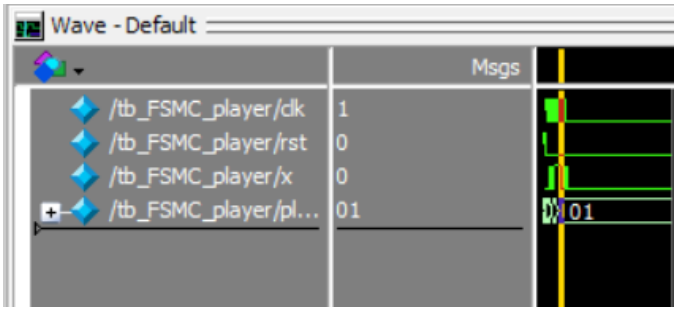

Figura 19. Testbench de la máquina de estados para cambiar de jugador.

de asignar una posición aleatoria de 1 a 9, sobre la posición que el sistema le asigna al jugador.

A su vez el módulo principal que contiene la lógica del juego está compuesto por otros módulos como:

- Contador: este se encarga de cambiar de posición buscando el jugador en turno la posición deseada para realizar jugada, su funcionamiento es correcto ya que como se comprobó en las pruebas realizadas realiza su función.
- Maquina estados cambiar jugador: en esta se diseño la lógica para realizar los cambios de turno entre un jugador y otro, su funcionamiento no es del todo correcto ya que en ocasiones al iniciar una partida realiza una doble jugada para el jugador inicial.
- Memoria de las casillas: en este modulo se implemento una memoria de registros para cada casilla del tic tac toe, el funcionamiento de este es el deseado ya que conserva la información de las jugadas realizadas previamente y a su vez no permite que se dibuje o se realice otra jugada sobre una casilla previamente utilizada.
- Seleccionar posición: la función de este modulo es que cuando el jugador se encuentre en la casilla deseada para realizar la jugada deseada, al seleccionar que se desea realizar en esa casilla funciona de acuerdo a lo planteado ya que con ayuda de otros módulos imprime la imagen deseada según el jugador de turno.
- Maquina de estados de la lógica del juego: esta maquina se encarga de realizar la conexión de señales entre los otros módulos que funcionan para la lógica del juego, su funcionamiento no es del todo el deseado ya que en ocasiones no conecta bien con el modulo de cambiar de jugador y le asigna un doble turno al jugador inicial, además no se encuentra conectada con el temporizador que determina la jugada random después el tiempo de espera de 30 segundos.

Por otro lado la implementación del controlador VGA funciona correctamente, ya que la sincronización con los ciclos de reloj y la carga de las imágenes desde un sprite en memoria, estas a su vez se instancia en un módulo dentro del controlador del VGA, finalmente se muestran en pantalla como se desea, sin embargo cabe recalcar que sin motivo conocido el fondo de las cuadrículas del tic tac toe brinca colores diferentes según la pantalla en la que se esté utilizando.

V. Conclusiones

Al final dicho laboratorio, se obtuvieron las siguientes conclusiones:

- Las máquinas de estado finito proporcionan una forma sistemática de diseñar circuitos secuenciales dada una función específica.
- Los controladores VGA aunque son fáciles de implementar presentan la desventaja de tener bajas resoluciones, además que la señal que emiten es de tipo analógica y al querer emitir señales digitales puede ocasionar problemas.
- Al momento de implementar soluciones complejas, las cuales se componen de varios componentes más pequeños y sencillos, lo óptimo es modularizar la propuesta, trabajando por medio del modelo estructural.
- Es necesario que todos los módulos del sistema operen bajo la misma señal de reloj, para este caso a 25MHz.
- Al crear máquinas de estado combinacionales, se puede comprobar mediante su correspondiente diagrama de estados, el adecuado funcionamiento de la misma y que corresponda al diseño original que fue desarrollado antes de programar la máquina.

Referencias

[1] Harris, S. Harris, D. *Digital Design and Computer Architecture: ARM Edition*. Morgan Kaufmann, 2015.

[2] *International Morse code Recommendation ITU-R M.1677-1*. itu.int. International Telecommunication Union. October 2009. Archived from the original on 6 November 2012. Recuperado el 27 de abril del 2021.

[3] Wolfram, S. *Starting From Randomness*. A New Kind of Science. Wolfram Media, p. 958. 2002. Recuperado el 27 de abril del 2021.

[4] Basharin G. et al. *The Life and Work of A. A. Markov*. Linear Algebra and its Applications 386: 3-26. 2004. Recuperado el 27 de abril del 2021.

[5] Fing.edu.uy. Técnicas de Manejo de E/S. Taller de Firmware. Facultad de Ingeniería. https://www.fing.edu.uy/inco/cursos/firmware/teorico/clase05-manejoES.pdf

[6] C. hardware?, J. Beckwith and S. Pefhany, Çan reading VGA signals from my computer harm the hardware?", Electrical Engineering Stack Exchange, 2021. Disponible: https://electronics.stackexchange.com/questions/166757/can-reading-vga-signals-from-my-computer-harm-the-hardware. Recuperado el 27 de abril del 2021..