Tarea 2 INF-245:

Acumulador de victorias entre competidores mediante circuitos secuenciales

Integrantes:

- **Claudio Inal ROL: 201873060-2**
- **Iván Cano ROL: 202073543-3**

Resumen

En este informe se dará a conocer el desarrollo de un circuito secuencial que es capaz de contar la cantidad de victorias entre dos competidores incluso variando entre rondas. Para ello se hará el uso de la lógica de una máquina de estados finitos Moore, que llevará un registro de las victorias hasta el momento para hacer las transiciones, la información de esta máquina será llevada a una tabla de verdad codificada de acuerdo al problema, de donde se extraerán funciones minimizadas mediante mapas de Karnaugh. Estas funciones serán bloques de un circuito combinacional en Logisim.
Adicionalmente justificamos el uso del circuito de la tarea anterior y un circuito combinacional auxiliar, los cuales juntos con las funciones antes mencionadas harán parte de la lógica de transición en nuestro circuito secuencial.
Para obtener la naturaleza secuencial en el circuito, haremos uso de Flip-Flops tipo D, que llevarán registro de la información pertinente. Estos Flip-Flops serán implementados según la materia del curso.
Finalmente se ensamblará el circuito en Logisim, el cual al ser testeado bajo diferentes pruebas se obtuvieron los siguientes resultados explicados posteriormente:

$$Éxitos\ del\ contador\ = \frac{64}{64} = 100\%$$

$$Éxitos\ en\ cambios\ de\ contrincante\ = \frac{2}{2} = 100\%$$

$$Éxitos\ en\ cambios\ de\ competidor\ fijo\ = \frac{2}{2} = 100\%$$

Introducción

Como se mencionó en la sección anterior, daremos uso de una máquina de estados finitos (FSM) Moore bajo diversas reglas de transición, las salidas de esta máquina se encuentran en el label de los estados y representan la cantidad de victorias acumuladas en un enfrentamiento entre dos competidores arbitrarios, durante una cantidad arbitraria de rondas consecutivas.

Luego se definirán las funciones que serán parte de la lógica de transición, una de ellas corresponde al circuito que se consiguió obtener en la tarea anterior, la cual define un ganador entre dos contrincantes. La segunda función es una función auxiliar capaz de comparar dos números de 3 bits cada uno.

Definidas estas funciones, pasamos a la elaboración de la lógica de transición como tal, que se encapsula en 4 diferentes funciones las cuales representan los bits de salida, es decir el contador de victorias. Para ello haremos las tablas de verdad para cada función, tendremos 6 variables en cada una: la primera corresponde al circuito de la tarea anterior, la segunda a la función auxiliar para comparar 2 números, y las últimas 4 a los estados anteriores a la transición. Una vez obtenidas estas funciones, crearemos un Flip-Flop tipo D con reseteo, extendido a 7 bits para tener un registro adecuado al problema, controlado por un reloj que es parte de Logisim.

Finalmente se ensamblará el circuito en Logisim con todos los bloques descritos hasta el momento, preparado para una serie de pruebas para validar su funcionamiento.

Desarrollo

Para comenzar haremos la máquina de estados finitos correspondiente al problema dado, el cual debe ser capaz de representar todas las victorias acumuladas posibles. Para esto estableceremos el label de nuestros estados como $q_i \forall i = 0, 1, \ldots, 15$, es decir, tendremos 16 estados diferentes. Para cada transición (arcos), primero estableceremos qué es lo que influye en los cambios de estados, para ello definiremos las siguientes funciones:

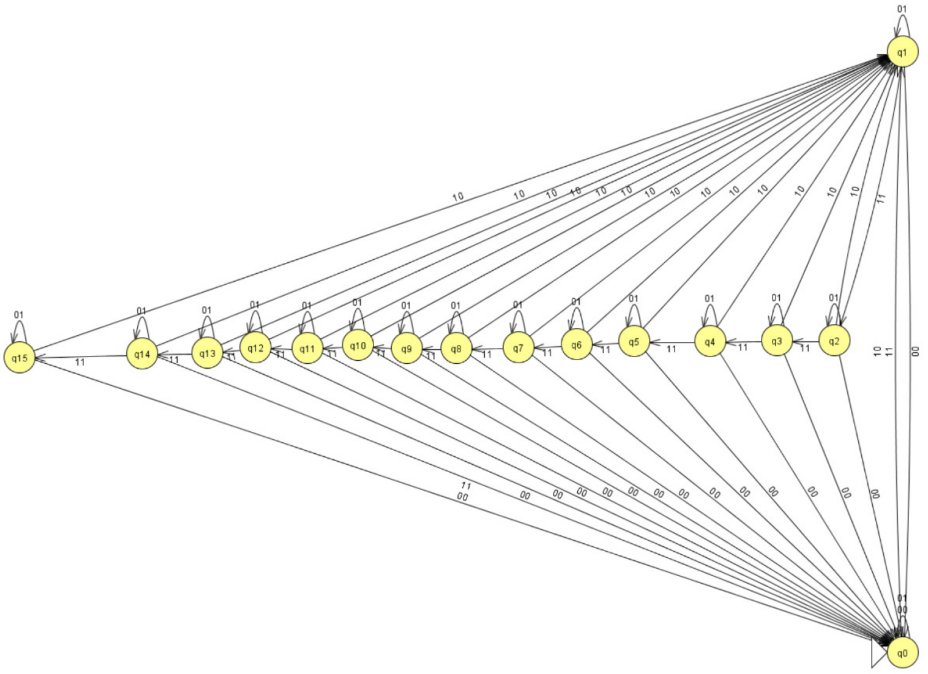
- $G(Competidor\ fijo\ actual,\ Contrincante)$:
  - 1, si $Competidor\ fijo\ actual$ es el ganador.
  - 0, en otro caso.
- $M(Competidor\ fijo\ actual,\ Competidor\ fijo\ anterior)$:
  - 1, si $Competidor\ fijo\ actual = Competidor\ fijo\ anterior$
  - 0, en otro caso.

Donde el competidor fijo actual, es aquel que se encuentra en el pin A, el contrincante es aquel en el pin B, y el competidor fijo anterior es aquel que estaba antes de transicionar en un estado, en uno de los pines de salida del Flip-Flop D.
Para definir los label de los arcos haremos uso de G y M concatenados, es decir tenemos 4 posibles label que harán cambios de estado, cada uno con una regla de transición diferente. Si estamos en el estado q_i las reglas de transición son según la siguiente tabla:

G	M	Regla
0	0	$q_i \rightarrow q_0$, el competidor fijo perdió, y no es el mismo al anterior o i=15
0	1	$q_i \rightarrow q_i$, el competidor fijo perdió, y es el mismo al anterior
1	0	$q_i \rightarrow q_1$, el competidor fijo ganó, y no es el mismo al anterior
1	1	$q_i \rightarrow q_{i+1}$, el competidor fijo ganó, y es el mismo al anterior (i<15)

Tabla 1: reglas de transición entre un estado y otro.

Si en un principio tenemos el contador de victorias en cero ($i = 0$), nuestro estado inicial será q_0, desde ahí podemos bosquejar la máquina de estados finitos (Moore) de la siguiente manera:



Hasta ahora hemos asumido la existencia de G y M, pero ahora debemos definirlas para poder darles uso en el circuito secuencial. Afortunadamente ya tenemos G a disposición, y corresponde al circuito combinacional de la tarea anterior que, como la sigla sugiere, establece al ganador entre dos competidores. Para M debemos realizar un poco más de trabajo, la sigla M viene dada por “mismo”, y es que esta función solamente compara dos números, entregando 1 si son los mismos, y 0 en otro caso. Podemos construir esta función comparando solamente 2 bits, y extenderla adecuadamente según la siguiente tabla de verdad:

p	q	m
0	0	1
0	1	0
1	0	0
1	1	1

Tabla 2: tabla de verdad para comparar 2 bits.

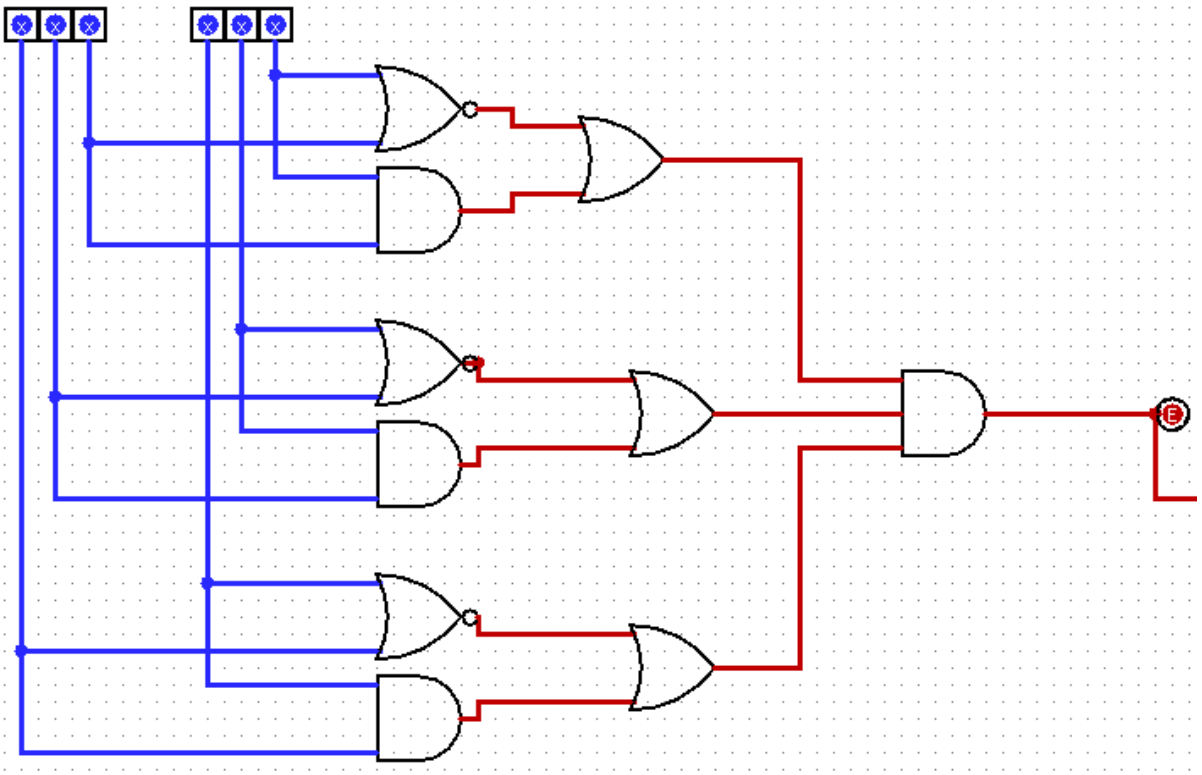
Por lo que m (para comparar 2 bits) corresponde a:

$$m(p,q) = \overline{p} \times \overline{q} + pq = \overline{p + q} + pq$$

Pero necesitamos que m pueda comparar 3 pares de bits, por lo que podemos extenderla como una productoria, así, será verdadero cuando todos los bits correspondientes sean iguales de la siguiente manera:

$$M = m(A0, B0) \times m(A1, B1) \times m(A2, B2)$$

Entonces el circuito combinacional de este bloque será de la siguiente manera:



Ya definidas satisfactoriamente las funciones G y M, podemos empezar a darles uso. Comenzaremos por mostrar las tablas de verdad: primero tendremos el estado actual S codificado en binario (entre 0 y 15), luego los inputs de GM codificado en binario (entre 0 y 3), y luego los estados siguientes S' codificados en binario (entre 0 y 15).

S3	S2	S1	S0	G	M	S'3	S'2	S'1	S'0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0

S3	S2	S1	S0	G	M	S'3	S'2	S'1	S'0
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0	1
0	0	1	0	0	1	0	0	1	0
0	0	1	1	0	1	0	0	1	1
0	1	0	0	0	1	0	1	0	0
0	1	0	1	0	1	0	1	0	1
0	1	1	0	0	1	0	1	1	0
0	1	1	1	0	1	0	1	1	1
1	0	0	0	0	1	1	0	0	0
1	0	0	1	0	1	1	0	0	1
1	0	1	0	0	1	1	0	1	0
1	0	1	1	0	1	1	0	1	1
1	1	0	0	0	1	1	1	0	0
1	1	0	1	0	1	1	1	0	1
1	1	1	0	0	1	1	1	1	0
1	1	1	1	0	1	1	1	1	1

Tabla 3 y 4: estado actual, inputs y estado siguiente según G y M.

S3	S2	S1	S0	G	M	S'3	S'2	S'1	S'0
0	0	0	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0	0	1
0	0	1	0	1	0	0	0	0	1
0	0	1	1	1	0	0	0	0	1
0	1	0	0	1	0	0	0	0	1
0	1	0	1	1	0	0	0	0	1
0	1	1	0	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1
1	0	0	0	1	0	0	0	0	1
1	0	0	1	1	0	0	0	0	1
1	0	1	0	1	0	0	0	0	1
1	0	1	1	1	0	0	0	0	1
1	1	0	0	1	0	0	0	0	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	0	0	0	1
1	1	1	1	1	0	0	0	0	1

S3	S2	S1	S0	G	M	S'3	S'2	S'1	S'0
0	0	0	0	1	1	0	0	0	1
0	0	0	1	1	1	0	0	1	0
0	0	1	0	1	1	0	0	1	1
0	0	1	1	1	1	0	1	0	0
0	1	0	0	1	1	0	1	0	1
0	1	0	1	1	1	0	1	1	0
0	1	1	0	1	1	0	1	1	1
0	1	1	1	1	1	1	0	0	0
1	0	0	0	1	1	1	0	0	1
1	0	0	1	1	1	1	0	1	0
1	0	1	0	1	1	1	0	1	1
1	0	1	1	1	1	1	1	0	0
1	1	0	0	1	1	1	1	0	1
1	1	0	1	1	1	1	1	1	0
1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0

Tabla 5 y 6: estado actual, inputs y estado siguiente según G y M. (notar que el estado en rojo no se utiliza)

Haremos los mapas de Karnaugh de las funciones S', para luego minimizarlas a una función que nos permita implementarlas en un bloque de circuito combinacional:

	S3S2	$\overline{M}$				M			
S1S0		00	01	11	10	00	01	11	10
$\overline{G}$	00	0	0	0	0	0	1	1	0
	01	0	0	0	0	0	1	1	0
	11	0	0	0	0	0	1	1	0
	10	0	0	0	0	0	1	1	0
G	00	1	1	1	1	1	0	0	1
	01	1	1	1	1	1	0	0	1
	11	1	1	1	1	1	0	0	1
	10	1	1	1	1	1	0	0	1

	S3S2	$\overline{M}$				M			
S1S0		00	01	11	10	00	01	11	10
$\overline{G}$	00	0	0	0	0	0	0	1	1
	01	0	0	0	0	0	0	1	1
	11	0	0	0	0	0	0	1	1
	10	0	0	0	0	0	0	1	1
G	00	0	0	0	0	0	1	0	1
	01	0	0	0	0	0	1	0	1
	11	0	0	0	0	0	1	0	1
	10	0	0	0	0	0	1	0	1

Tabla 7 y 8: K-Map de las funciones S'0 y S'1 respectivamente.

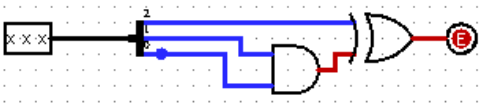
	S3S2	$\overline{M}$				M			
S1S0		00	01	11	10	00	01	11	10
$\overline{G}$	00	0	0	0	0	0	0	0	0
	01	0	0	0	0	1	1	1	1
	11	0	0	0	0	1	1	1	1
	10	0	0	0	0	0	0	0	0
G	00	0	0	0	0	0	0	1	0
	01	0	0	0	0	1	1	0	1
	11	0	0	0	0	1	1	0	1
	10	0	0	0	0	0	0	1	0

	S3S2	$\overline{M}$				M			
S1S0		00	01	11	10	00	01	11	10
$\overline{G}$	00	0	0	0	0	0	0	0	0
	01	0	0	0	0	0	0	0	0
	11	0	0	0	0	1	1	1	1
	10	0	0	0	0	1	1	1	1
G	00	0	0	0	0	0	0	0	0
	01	0	0	0	0	0	0	1	0
	11	0	0	0	0	1	1	0	1
	10	0	0	0	0	1	1	1	1

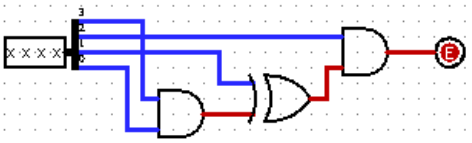
Tabla 9 y 10: K-Map de las funciones S'2 y S'3 respectivamente.

Finalmente conseguimos las siguientes funciones minimizadas con su respectivo circuito:

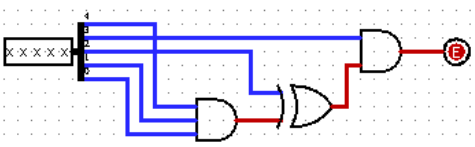
- $S'_0 = G \oplus MS_0$:



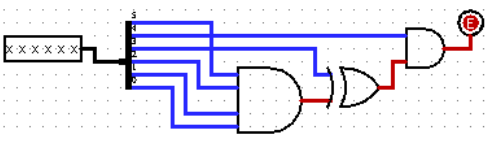
- $S'_1 = M(S_1 \oplus GS_0)$:



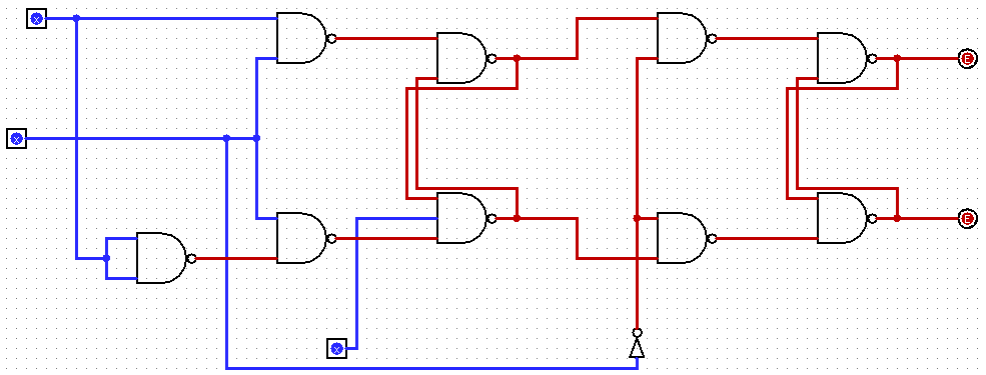
- $S'_2 = M(S_2 \oplus GS_1S_0)$:



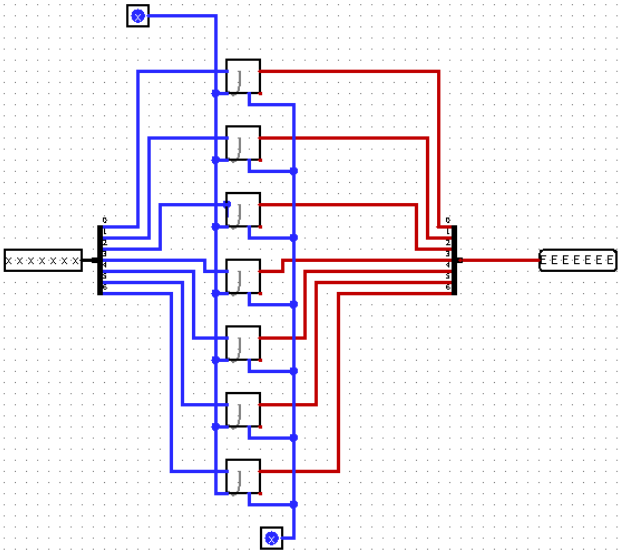
- $S'_3 = M(S_3 \oplus GS_2S_1S_0)$:



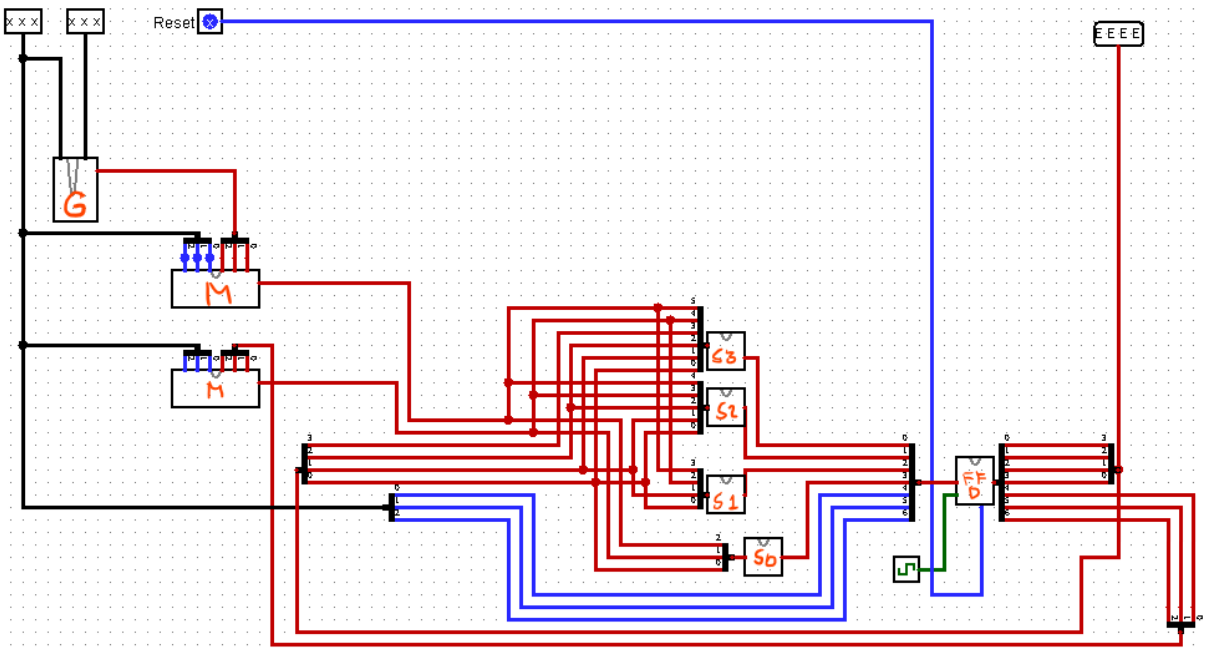
El componente del circuito que nos permitirá hacerlo secuencial es un Flip-Flop tipo D con un reset para poder establecer el estado inicial, tal como muestra el circuito:



Luego unimos en paralelo 7 de estos FF-D para tener un registro de 7 bits, en donde los primeros 4 (0, 1, 2 y 3) corresponden las victorias acumuladas, y los últimos 3 (4, 5 y 6) será el competidor fijo anterior, como se muestra en el circuito:



Teniendo ya todos los componentes podemos finalmente implementar el circuito secuencial:



Resultados

El primer test a realizar, similar a la tarea anterior, será el de verificar que los competidores fijos ganen o pierdan siempre en 15 etapas, sin importar la cantidad de victorias acumuladas, es decir, si un competidor fijo siempre pierde contra el contrincante desde la etapa 1 hasta la 15, el contador de victorias será 0, y si siempre gana, después de 15 etapas el contador de victorias será 15. Ilustraremos lo anterior con la siguiente matriz, en donde el competidor fijo es i (fila) y el contrincante es j (columna) en ese orden:

ID	0	1	2	3	4	5	6	7
0	15	0	0	0	15	0	0	0
1	15	15	0	15	15	15	0	0
2	15	15	15	15	15	15	0	0
3	15	0	0	15	15	0	0	0
4	0	0	0	0	15	0	0	0
5	15	0	0	15	15	15	0	0
6	15	15	15	15	15	15	15	15
7	15	15	15	15	15	15	0	15

Tabla 11: Matriz de victorias acumuladas según los competidores de la ronda después de 15 rondas.

El segundo test a realizar será el de, teniendo un competidor fijo y un contrincante contra el que gana, cambiar el contrincante por otro en donde pierda, esto debería empezar a sumar una victoria después de una ronda. Haremos también lo inverso, es decir cambiar un contrincante contra el que siempre gana, por uno donde siempre pierda, deteniéndose el contador de victorias. Al haber tantas combinaciones posibles solo haremos 1 por cada caso:

- Caso 1:
  - C. fijo: 0, C. ronda: 1, Victorias después de una etapa: 0
  - C. fijo: 0, C. ronda: 4, Victorias después de dos etapas: 1 (aumentó)
- Caso 2:
  - C. fijo: 0, C. ronda: 4, Victorias después de una etapa: 1
  - C. fijo: 0, C. ronda: 1, Victorias después de dos etapas: 1 (no cambió)

El tercer test a realizar será el de cambiar el competidor fijo. Tenemos 2 casos diferentes, el primero es si al cambiar el competidor fijo, queda una combinación en donde siempre pierde, el contador de victorias vuelve a 0. El segundo caso es si al cambiar el competidor fijo, llegamos a una combinación en donde siempre gana, el contador de victorias vuelve a 1, no a 0. Al haber tantas combinaciones posibles solo haremos 1 por cada caso:

- Caso 1:
  - C. fijo: 7, C. ronda: 5, Victorias después de doce etapas: 12
  - C. fijo: 4, C. ronda: 5, Victorias después de una etapa: 0 (vuelve a 0)
- Caso 2:
  - C. fijo: 7, C. ronda: 3, Victorias después de diez etapas: 10
  - C. fijo: 6, C. ronda: 3, Victorias después de una etapa: 1 (vuelve a 1)

Análisis

Observando los resultados de la sección anterior, notamos un 100% de éxitos (marcados en verde) para los tests dispuestos en la capacidad de acumular (o contar) las victorias entre un competidor fijo y otro competidor arbitrario siempre y cuando este gane, como fue dispuesto en la matriz:

$$Éxitos\ del\ contador\ = \frac{64}{64} = 100\%$$

También podemos notar un 100% de éxitos en las transiciones entre competidores de ronda, ya se deteniendo el contador si pierde, o aumentando si es que gana:

$$Éxitos\ en\ cambios\ de\ contrincante\ = \frac{2}{2} = 100\%$$

Finalmente notamos un 100% de éxitos en las transiciones entre competidores fijos, ya sea volviendo a 0 si es que pierde, y a 1 si es que gana:

$$Éxitos\ en\ cambios\ de\ competidor\ fijo\ = \frac{2}{2} = 100\%$$

A pesar de la baja cantidad de casos realizadas para los tests 2 y 3, podemos justificar esto declarando haber elegido los casos de forma aleatoria, y es que si hubiésemos hecho la búsqueda exhaustiva, tendríamos una cantidad enorme de combinaciones: podemos elegir 8 competidores fijos iniciales, 7 competidores de ronda, y si cambiamos de competidor de ronda después de cada etapa, tendríamos un total de 8*7*15 casos diferentes para el test 2, lo cual resulta tedioso probar a mano. Un análisis similar se hace para el test 3, en donde se cambia el competidor fijo.

Conclusión

El acumulador de victorias es capaz de contar hasta el máximo posible sin errores cuando enfrentamos dos competidores consecutivamente (dependiendo de quién gana), aunque también dimos la posibilidad de resetear el contador a 0 una vez supere el límite de 15, esto para ser consistentes con la máquina de estados finitos, y no perdernos una transición que podría incurrir en errores en el circuito (a pesar de que se mencionó que no se probarán más de 15 etapas).

Los cambios de competidores fijos y competidores de rondas también fueron un éxito, a pesar de las limitantes de tiempo para realizar una búsqueda exhaustiva entre todos los casos posibles (como se justificó en la sección anterior).

El nivel de finalización de la tarea a nuestro criterio es total. No se ahondó en procedimientos algorítmicos (como la extracción de funciones minimizadas en los mapas de Karnaugh) o la creación de la máquina de estados finitos para mantener orden y legibilidad en el informe.