| | |
|---|---|
| - Nombre y CI en cada hoja<br>- Numere las hojas<br>- Indique el total de hojas en la primera<br>- Utilice solo UN lado de las hojas | - Incluya un solo problema por hoja<br>- **Sea prolijo**<br>- **Aprobación:** mínimo UN problema y UN ejercicio completos |

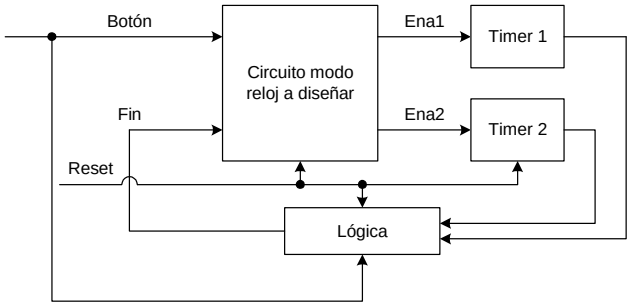
**PROBLEMA 1**

Se quiere usar un dispositivo con doble cronómetro que totalice el tiempo invertido por dos jugadores en una partida. Este dispositivo funciona de la siguiente forma: cada vez que un jugador oprime la entrada **Botón** se detiene el cronómetro que cuenta su tiempo y comienza a funcionar el cronómetro que cuenta el tiempo de su rival.

Ambos cronómetros nunca funcionarán en forma simultánea. Los jugadores tendrán un límite de tiempo para el total de las jugadas de la partida.

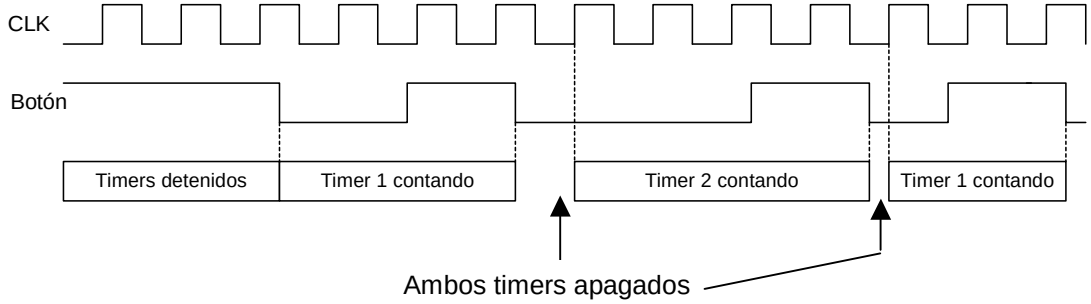
Se desea diseñar un circuito **modo reloj** que implemente este dispositivo y cumpla con las siguientes especificaciones:

- Entradas :
  - **Botón**: para que los jugadores puedan detener su cronómetro e iniciar el de su contrincante. Se activa por bajo cuando es presionado y vuelve a uno cuando lo sueltan, se supone que no tiene rebotes y que la duración de los pulsos es mayor que un período de reloj.
  - **Fin** : indica que alguno de los jugadores agotó su tiempo. Activa por alto.
- Salidas :
  - **Ena1** : habilitación del cronómetro del jugador 1, activa por nivel alto
  - **Ena2** : habilitación del cronómetro del jugador 2, activa por nivel alto
- El sistema tendrá una entrada asíncrona **Reset** que llevará al dispositivo a un estado de reposo y reseteará los cronómetros de los jugadores.
- Luego de un reset las entradas **Fin** y **Botón** valen 1 e inmediatamente después que **Botón** es presionado **Fin** pasa a valer 0.
- Cuando se presiona la entrada **Botón** por primera vez, empieza a correr el tiempo del jugador 1 inmediatamente.




- Luego de esta situación inicial, cuando un jugador presiona **Botón** su cronómetro se detendrá en ese instante y el cronómetro de su adversario se iniciará en el flanco de reloj siguiente.
- Cuando se acaba el tiempo total de un jugador la entrada **Fin** pasa a 1 y el sistema debe ir al estado de reposo inicial, no importando ya el valor que tomen las salidas **Ena1** y **Ena2** dado que **Fin** queda en 1 hasta un nuevo reset, independientemente de las mismas.
- Para iniciar una nueva partida debe activarse **Reset** nuevamente.

En la siguiente figura se muestra un ejemplo de cómo debe funcionar el circuito:



**PROBLEMA 2**

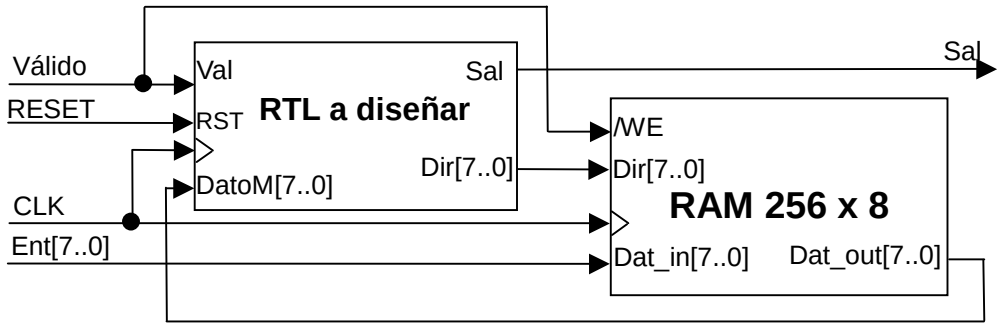
Se desea diseñar el circuito, para recibir ráfagas de datos de 8 bits en paralelo y transmitirlos en formato serie. Por ráfagas, se entiende que se reciben muchos datos seguidos durante un tiempo, luego se da un reposo por otro lapso de tiempo antes de volver a recibir, y así sucesivamente. Durante la ráfaga, los datos pueden llegar en forma consecutiva (un dato por período de reloj) o espaciada (un dato cada algunos períodos de reloj).

Los datos se reciben en forma sincronizada con el flanco creciente de reloj. La entrada ***Ent[7..0]*** es válida si la entrada ***Valido*** = 0. En caso contrario ***Ent[]*** no está determinado. Si los datos son consecutivos se reciben con ***Valido*** = 0 y ***Ent[]*** cambia cada período de reloj.

Los datos deben enviarse en forma serie por la salida ***Sal***, comenzando con 1 bit de arranque en nivel bajo, 8 bits de datos (primero el MSB) y finalmente 1 bit de parada en nivel alto. ***Sal*** debe estar sincronizada con el flanco creciente de reloj. Cuando no se transmite **Sal** = 1.

Para soportar las ráfagas, los datos se deben almacenar temporalmente en una cola circular FIFO (First In First Out -- primero que ingresa primero que sale) implementada en una memoria RAM síncrona con entrada y salida de datos independientes, que funciona de la siguiente forma. En cada flanco de subida de reloj, si ***/WE*** = 0, el dato presente en ***Dat_in[]*** se copia en la dirección indicada por ***Dir[]***. En cambio si ***/WE*** = 1 , el dato en la dirección ***Dir[]*** se copia a la salida ***Dat_out***. Si ***/WE*** = 0, ***D_out[]*** no está definida.

Se debe diseñar el circuito RTL con las entradas y salidas indicadas en la figura. Para implementar la cola circular, se deben definir 2 registros, **Dir_in[]** y **Dir_out[]** de forma que inicialmente ambos registros sean nulos. Para poner un dato en la cola se debe hacer en la dirección **Dir_in[]**, incrementando luego su valor. Para quitar un dato de la cola se debe hacer en la dirección **Dir_out[]**, también incrementando luego su valor. Tener presente que si **Dir_in[]** = **Dir_out[]** implica que no hay datos en la cola. Se supondrá que la cola nunca llega a llenarse.




Consideraciones:

Mientras existan datos en la cola, se deben enviar a la salida en forma serial.

Se debe contemplar que puede venir un dato válido mientras se está transmitiendo.

## EJERCICIO 1

a) Codificar en Hamming de 7bits: m3 m2 m1 m0 = 0100

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| P1 | P2 | m3 | P3 | m2 | m1 | m0 |

b) La siguiente es una palabra de 4 bits codificada en código Hamming de 7 bits: 1010111
Indicar cual era la palabra de 4 bits original.

## EJERCICIO 2

Las siguientes ecuaciones corresponden a un circuito modo nivel:

y1 = !A.B + !A.y1 + B.y1
y0 = A.!B +!B.y0 + A.y1
S = A + !y1.y0 + !B. !y1

y1 e y0 son las variables de estado y S la salida.

Se pide:

a) Mapas K y tabla de estados.
b) Determinar si presenta carreras y de qué tipo. Justificar.