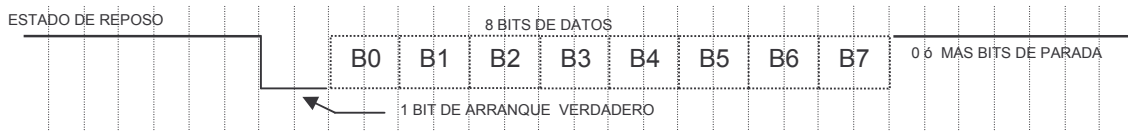


- Mínimo de aprobación: un problema y un ejercicio
- Cada hoja debe tener Nombre y CI.
- Utilice solo un lado de las hojas
- Deben estar numeradas y la primer hoja debe decir el total de hojas
- Incluya un solo problema por hoja
- Sea prolijo

### Problema 1

Se reciben datos de 8 bits por medio de una señal serial que se encuentra sincronizada con el reloj del sistema. La señal consta de un bit de arranque, 8 bits de datos (comenzando por el bit menos significativo) y no necesariamente hay bits de parada, donde cada bit demora 2 períodos de reloj. Puede pasar que existan falsos bit de arranque, los que valen 0 en el primer período de reloj y 1 en el segundo, y luego no hay transmisión de datos.



Se debe diseñar un circuito con entrada **IN** y salidas **DATO[7..0]** y **RDY**, que sea capaz de convertir el dato recibido de serie a paralelo. Este circuito debe constar de un circuito **modo reloj a diseñar**, un registro de corrimiento (shift register) de 8 bits, un contador de 4 bits y una condición lógica a definir.

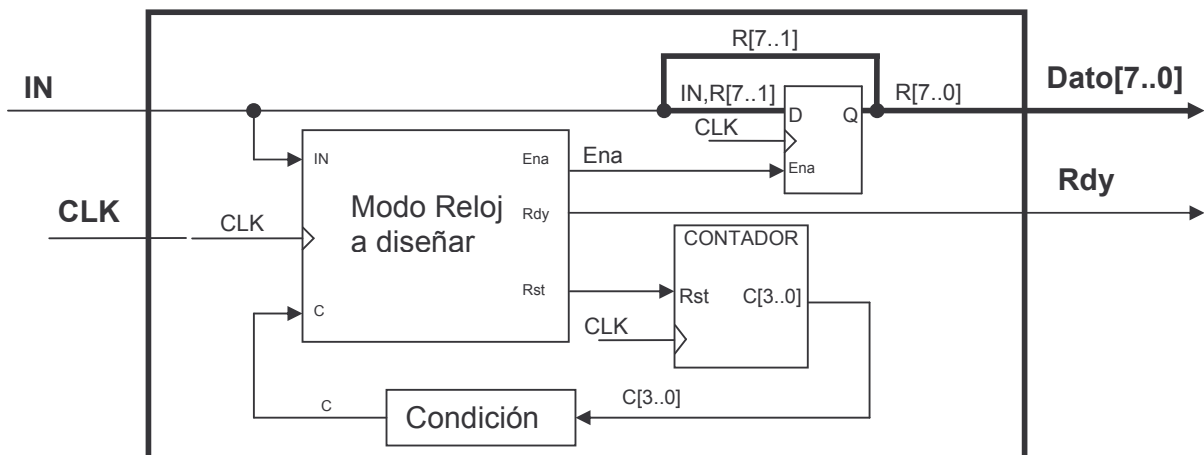
El registro de corrimiento **R[7..0]** solo responde al flanco creciente de reloj si **Ena** = 1.

El contador, es un contador circular que dispone de una entrada **Rst**. Si **Rst** = 1, **C[3..0]** = 0000, en cambio, si **Rst** = 0, el contador incrementará **C[3..0]** en 1 en cada flanco creciente de reloj.

La condición lógica es un circuito combinatorio a definir en forma consistente con el circuito modo reloj (por ej podrá indicar cuando **C[3..0]** es igual, distinto, mayor, etc a un valor dado).

Se pide:

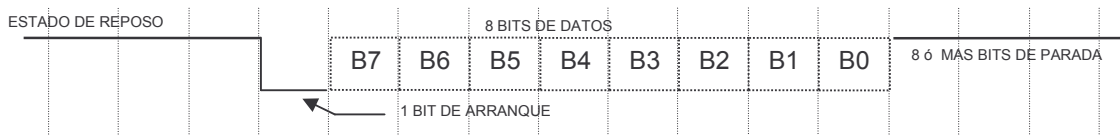
- definir claramente con un circuito combinatorio el bloque “Condición”.
- diseñar el circuito **Modo Reloj** de forma que el circuito completo se comporte como:
  - Para el caso de un *falso bit de arranque*:
    - se debe ignorar la lectura de datos.
    - la señal **Rdy** debe valer 0 desde que baja hasta que sube **IN**.
    - el circuito debe estar pronto para recibir un nuevo bit de arranque a continuación.
  - Para el caso de un *verdadero bit de arranque*:
    - las lecturas de los bits de datos se deben realizar en el centro de cada bit.
    - la señal **Rdy** debe valer 0 desde que baja **IN**, hasta la mitad del último bit de datos (al leer este último bit).
    - el circuito debe estar pronto para recibir un nuevo bit de arranque a continuación del último bit de dato.
  - Una vez en funcionamiento, el circuito se supondrá siempre sincronizado con la señal **IN**.



## Problema 2

Se desea automatizar la velocidad de un tren a lo largo de todo su recorrido controlando la señal, **Acelerador[8]**, de forma que el tren aumente o disminuya su velocidad en forma autónoma.

El tren dispone de un velocímetro digital que, cada cierto tiempo, transmite la velocidad actual del tren por su salida **Vel\_actual** sincronizada con el reloj del sistema. La transmisión es en formato serie, con 1 bit de arranque, 8 bits de datos (comenzando por el MSB) y al menos 8 bits de parada.



El tren posee además un escáner que va leyendo durante el recorrido emisores ubicados junto a la vía, que indican la velocidad óptima (**Vel\_op**) y máxima (**Vel\_max**) de circulación. Este escáner tiene 3 salidas: **Válido**, **Vel\_op[8]** y **Vel\_max[8]**. Cada vez que el escáner captura nuevos datos, **Válido** sube a 1 durante un período de reloj para indicar que hay datos válidos en **Vel\_op[8]** y en **Vel\_max[8]**.

Se pide diseñar un circuito RTL conectado a estos 2 dispositivos, para controlar la señal **Acelerador[8]**, y que funcione como se explica a continuación.

Cuando el maquinista prende el automatismo, primero se deberá esperar a recibir un dato del escáner y luego esperar 8 bits de parada consecutivos en el velocímetro (se debe descartar toda lectura de velocidad en tanto no se comprueben los 8 bits de parada). En tanto esto no ocurra, el **Acelerador** debe estar en su valor máximo (1111 1111).

A partir de esto, cada vez que se lea **Vel\_actual** se debe modificar **Acelerador** de forma que si **Vel\_actual** es inferior o igual a **Vel\_op**, se debe aumentar **Acelerador** en 1, si es que no se encuentra en su valor máximo (1111 1111). Si **Vel\_actual** se encuentra entre **Vel\_op** y **Vel\_max**, se debe disminuir **Acelerador** en 1, si es que no se encuentra en su valor mínimo (0000 0000). Para el caso en que la **Vel\_actual** sea superior o igual a la **Vel\_max**, se debe desacelerar al mínimo (**Acelerador** = 00000000).

Se dispone de bloques incrementadores, sumadores, decrementadores, restadores (tanto circulares como con saturación), comparadores, etc. Indicar correctamente el tipo de bloque que se utilice.

Se pide: descripción RTL del circuito, bloque de datos y bloque de control

## Ejercicio 1

Se desean transmitir nibbles (palabras de 4 bits) través de un canal con interferencias. Para eso se genera a partir de los mismos un código de Hamming de 7 bits utilizando paridad par. Decodifique el siguiente mensaje recibido, asumiendo que a lo sumo ha ocurrido un único error en cada palabra del código: 0011001 0101001 1110110

(Orden de los bits: p1 p2 m3 p3 m2 m1 m0 Nibble: m3 m2 m1 m0)

## Ejercicio 2

Dada la siguiente tabla de un circuito secuencial modo nivel, se pide:

	00	01	11	10
Q0	Q2	<b>Q0/1</b>	Q1	-
Q1	Q3	Q0	<b>Q1/0</b>	<b>Q1/0</b>
Q2	<b>Q2/0</b>	Q0	<b>Q2/1</b>	Q1
Q3	<b>Q3/1</b>	Q0	Q1	<b>Q3/1</b>

- Encontrar una asignación sin carreras sin agregar variables de estado.
- Asignar salidas para que no haya espurios.