- Mínimo de aprobación: **un problema y un ejercicio completos**
- 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 quiere diseñar un circuito modo nivel que funcione como divisor de frecuencia por 2 de la señal **Onda** y cuente con entrada de habilitación **Ena**.

En régimen cuando la entrada **Ena** =1 el circuito debe copiar en **Z** los pulsos a uno de **Onda,** suprimiendo un pulso por medio. De esa forma la frecuencia de **Z** será la mitad que la frecuencia de **Onda**.

También en régimen cuando **Ena** = 0, la salida **Z** deberá valer cero.

Se desea que nunca queden pulsos en **Z** más cortos que los de la entrada **Onda**, por lo tanto:

- si **Ena** sube cuando **Onda** = 1, deberá esperarse a que **Onda** baje y vuelva a subir para empezar a copiar el primer pulso en **Z**.
- si **Ena** baja cuando **Onda** =1 y **Z**=1, deberá mantenerse la salida **Z**=1 hasta que baje **Onda**

Si **Ena** baja y vuelve a subir mientras **Onda** =1 entonces se debe mantener la salida **Z**=1.

Diseñar un circuito modo nivel sin carreras ni espurios en la salida.

Problema 2

Se reciben por una señal **Ent[7..0]** paquetes de 256 bytes de información en forma desordenada. Para poder ordenarlos y que la información tenga sentido, cada byte viene seguido de un *índice* que es un número de 8 bits que indica la posición que le corresponde al byte de información recibido. **Ent[7..0]** es tal que:

- **Ent[7..0]** está sincronizada con el flanco de bajada del reloj del sistema.
- Tanto el byte de información, como el índice duran 1 período de reloj.
- Si **Ent[7..0]** = 0x00 indica que no hay datos válidos. Un byte de información difiere de 0x00.
- Cuando **Ent[7..0]** pasa a ser distinto de 00h, se trata de un byte de información válido.
- Cada byte de información es seguido del *índice* en el siguiente período de reloj.
- Si a continuación del *índice*, **Ent[7..0]** <> 00h, indica un nuevo byte de información.
- Cada byte de información puede tener cualquier índice.

Se desea generar una salida **Sal[7..0]** tal que:

- **Sal[7..0]** esté sincronizada con el flanco de bajada del reloj del sistema.
- **Sal[7..0]** = 00h indique que no hay datos válidos.
- Los bytes de información deben enviarse en forma ordenada y durar un período de reloj.
- El primer byte a transmitir por **Sal[]** debe ser el que tiene índice=00h, y luego el que tiene índice=01h, 02h, y así sucesivamente hasta llegar al FFh.
- No se transmite índice ya que no es necesario.

Para poder almacenar temporalmente los bytes de información que van llegando y no corresponde transmitir, se debe utilizar una memoria RAM síncrona con entrada y salida de datos independientes, que funciona de la siguiente forma:

- El contenido almacenado en la dirección ***Dir[]*** se muestra en la salida ***Dat_out[]*** *en todo momento*.
- En cada flanco de bajada de reloj, si ***WR*** = 1, el dato presente en ***Dat_in[]*** se almacena en la dirección indicada por ***Dir[]***.
- La señal **RST** borra toda la memoria, quedando el valor 0x00 en todas las direcciones.

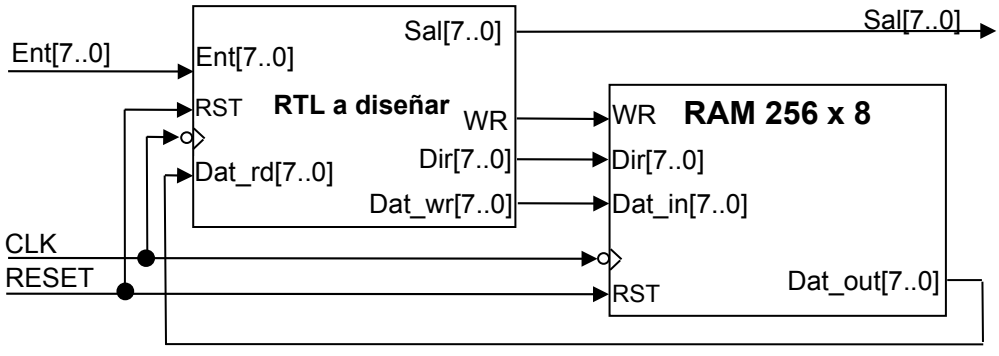
Luego de un reset se sabe que no se recibirán datos válidos por lo menos por un período de reloj, y lo primero que llega es un byte de dato seguido de su índice correspondiente.

Se requiere una solución que transmita los bytes a medida que vayan llegando, no es válido esperar a que llegue el paquete completo de 256 bytes antes de empezar a transmitir.
Se garantiza que un nuevo paquete no comienza hasta que se haya terminado de trasmitir el paquete anterior.

Siempre que **Ent[]** = 0x00 y existan datos consecutivos almacenados en la RAM a ser transmitidos por **Sal[]**, se debe realizar una transmisión por período de reloj.

Al leer un dato de la RAM para transmitirlo por **Sal[]**, se debe borrar (cargar 00h), para indicar que el dato fue transmitido.

Se debe diseñar el circuito RTL con las entradas y salidas indicadas en la figura.



Ejercicio 1

Se tiene un código de 6 bits con las siguientes palabras:

101110
001000
111001
010101

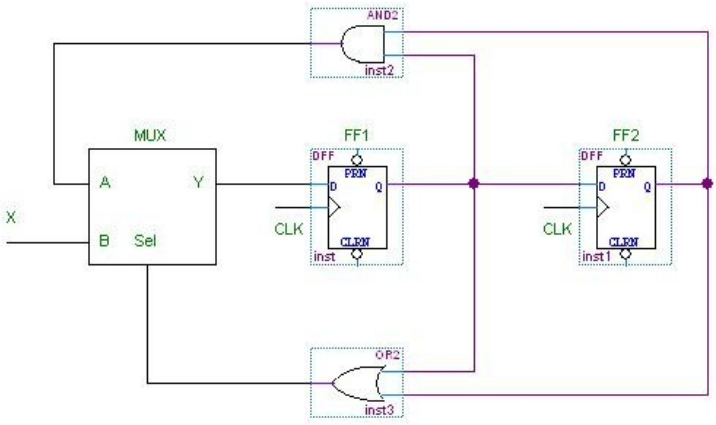
a) Indicar qué distancia tiene este código justificando. Indicar qué opciones se tienen en cuanto a detección y/o corrección de errores.

b) Para cada una de las opciones señaladas en a), indicar qué resultado se tendría al recibir las siguientes palabras:

i) 001001
ii) 000100

c) ¿ Qué opciones se tienen de detección y/o corrección de errores si se agrega un bit de paridad al código anterior ?

Ejercicio 2



a) Determinar la frecuencia máxima de funcionamiento del circuito. Suponer la entrada X constante. Justificar claramente con un diagrama de tiempo, indicando a qué FF corresponden los parámetros utilizados.

b) Indicar cuál es el intervalo de tiempo durante el cual debe permanecer constante X para asegurar un correcto funcionamiento. Tomar como referencia el flanco de subida de clock.

Datos:
Tiempo de setup de FF: tsmin < ts
Tiempo de hold de FF: thmin < th
Tiempo de propagación de FF: tpmin < tp < tpmax

Tiempo de prop. en compuertas and: tandmin < tand < tandmax
Tiempo de prop. en compuertas or: tormin < tor < tormax
Tiempo de prop. en mux:
Selector-salida: tssmin < tss < tssmax
Entrada-salida: tesmin < tes < tesmax