EJERCICIOS PROPUESTOS SISTEMAS DIGITALES II

SEGUNDO PARCIAL:

SD + MSA

ALGORITMOS DE BUSQUEDA

1.) Realizar el diseño de un **SISTEMA EFICIENTE DE DETECCIÓN DE DIGITOS**. El sistema recibe una trama de dígitos entre 0 -9 de forma aleatoria y permite detectar el número que más veces se repite en toda la trama.

SEÑALES:

- **Keypad.-** son 10 teclas que permiten el ingreso de los dígitos entre 0-9.
- **Ingreso Datos.-** Esta señal le indica al sistema que el usuario desea ingresar la secuencia y se debe validar que el ingreso de los dígitos no sea mayor a 256.
- **Fin Ingreso Datos.-** si el número de dígitos ingresados no es mayor a 256 el usuario puede dejar de ingresar los números en cualquier momento.
- **LedSec.-** Led indicador que el sistema está solicitando el ingreso de la trama de dígitos.
- **Start.-** Luego de ingresar correctamente los datos el sistema valida la tecla start para empezar a trabajar.
- **Display Patrón.-** indicará el mejor digito detectado.
- **Display Repeticiones.-** Indica la cantidad de veces que se repite el digito en toda la trama.

Ejemplo:	
Trama (con dígitos entre 0 - 9)	24563129563245298061230689725
Dígito que más se repite	2
Repeticiones del Número	6

PRESENTAR:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

NOTA: el pseudocódigo para detectar las veces que se repite cada dígito común es.

```
Inicializar RAMCnt=0
For i=0 to TamSecuencia
        CNT=0;
        For j=0 to TamSecuencia
                Ri=RAM(i);
                Rj=RAM(j);
                If (Ri=Rj)
                        Inc(CNT);
                Endif
        endFor
        If (RAMCnt(RAM(i))<CNT)
                RAMCnt(RAM(i))=CNT;
        Endif
EndFor
```

Código mejorado:

```
Inicializar RAMCnt=0
For i=0 to TamSecuencia
        CNT=0;
        if(RAMCnt(RAM(i))=0)
                For j=i to TamSecuencia
                        Ri=RAM(i);
                        Rj=RAM(j);
                        If (Ri=Rj)
                                Inc(CNT);
                        Endif
                endFor
                RAM(RAM(i))=CNT;
        Endif
EndFor
```

RECURSOS: https://goo.gl/hozZZM

2.) Realizar el diseño de un **SISTEMA EFICIENTE DE DETECCIÓN DE SECUENCIA DE 3 DIGITOS**. El sistema recibe una trama de dígitos entre 0 -9 de forma aleatoria y permite detectar la secuencia de 3 números que más veces se repite en toda la trama.

SEÑALES:

- **Keypad.-** son 10 teclas que permiten el ingreso de los dígitos entre 0-9.
- **Ingreso Datos.-** Esta señal le indica al sistema que el usuario desea ingresar la secuencia y se debe validar que el ingreso de los dígitos no sea mayor a 256.
- **Fin Ingreso Datos.-** si el número de dígitos ingresados no es mayor a 256 el usuario puede dejar de ingresar los números en cualquier momento.
- **LedSec.-** Led indicador que el sistema está solicitando el ingreso de la trama de dígitos.
- **Start.-** Luego de ingresar correctamente los datos el sistema valida la tecla start para empezar a trabajar.
- **Display Patrón.-** se indicarán tres displays el mejor patrón detectado.
- **Display Repeticiones.-** Indica la cantidad de veces que se repite la mejor trama detectada.

Ejemplo:	
Trama (con dígitos entre 0 - 9)	24563129563245298061230689725456 3
Patrón de 3 dígitos que más se repite	563
Repeticiones del Patrón	3

RECURSOS: https://goo.gl/y80JMV

3.) Diseñe un sistema digital que evalúe que estudiantes han aprobado la materia de sistemas digitales 2. Considere que tiene una memoria en la que se encuentran almacenadas las notas de las tres evaluaciones del semestre. Cada estudiante tiene un identificador que es un número entero entre 1 y 40. En la memoria la información se encuentra almacenada de la siguiente manera:

Dirección	Dato
0x00	ID_1
0x01	Nota_1
0x02	Nota_2
0x03	Nota_3
0x04	ID_2
0x05	Nota_1
0x06	Nota_2
0x07	Nota_3
...	...
...	...

DONDE:

- ID_X representa el identificador del estudiante (reemplaza al nombre en la lista)
- Nota_1 es la nota de primera evaluación
- Nota_2 es la nota de segunda evaluación
- Nota_3 es la nota de la tercera evaluación

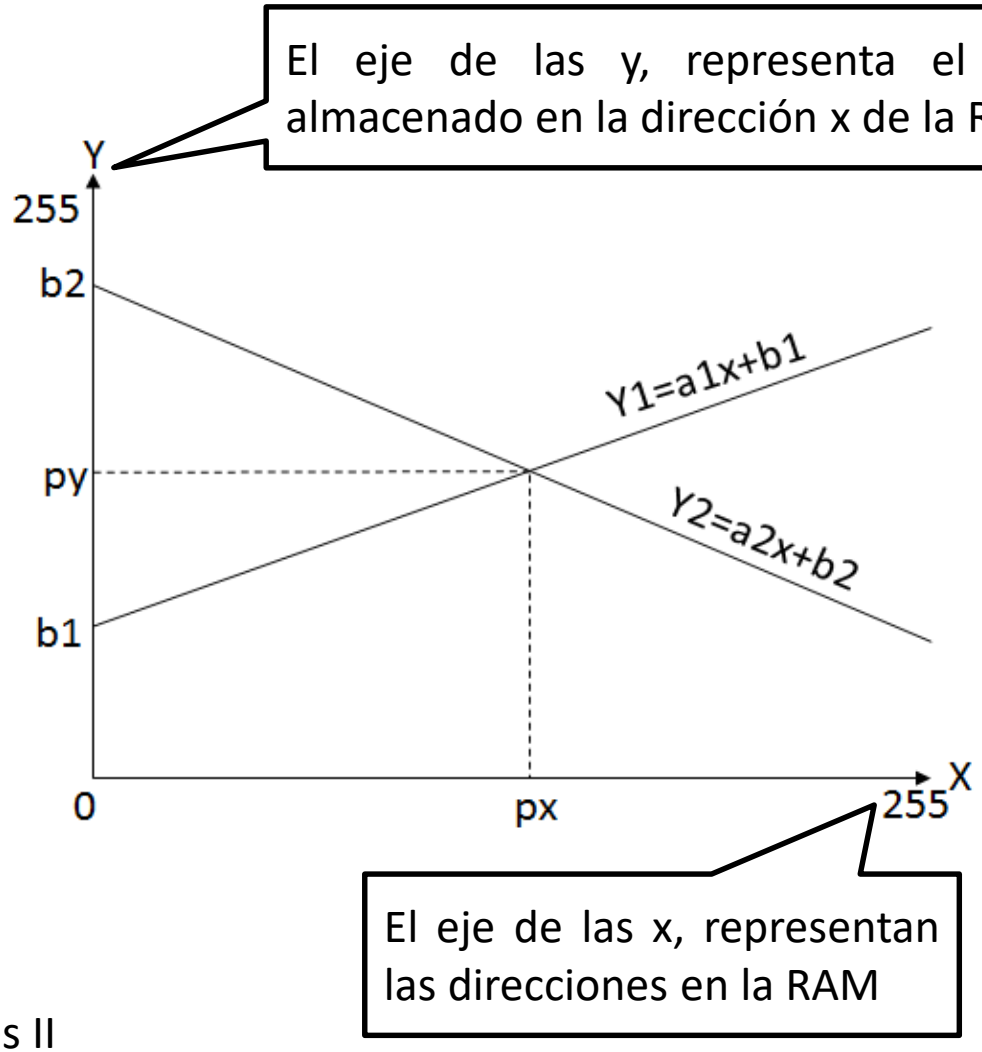
El sistema evalúa que estudiantes han aprobado y los muestra en displays de 7 segmentos en el siguiente orden: Primero muestra el ID del estudiante, espera 3 segundos y muestra el promedio con el que aprobó la materia y espera 5 segundos antes de mostrar los datos del siguiente estudiante aprobado. Cuando el sistema ha mostrado todos los estudiantes que aprobaron la materia genera la señal FIN.

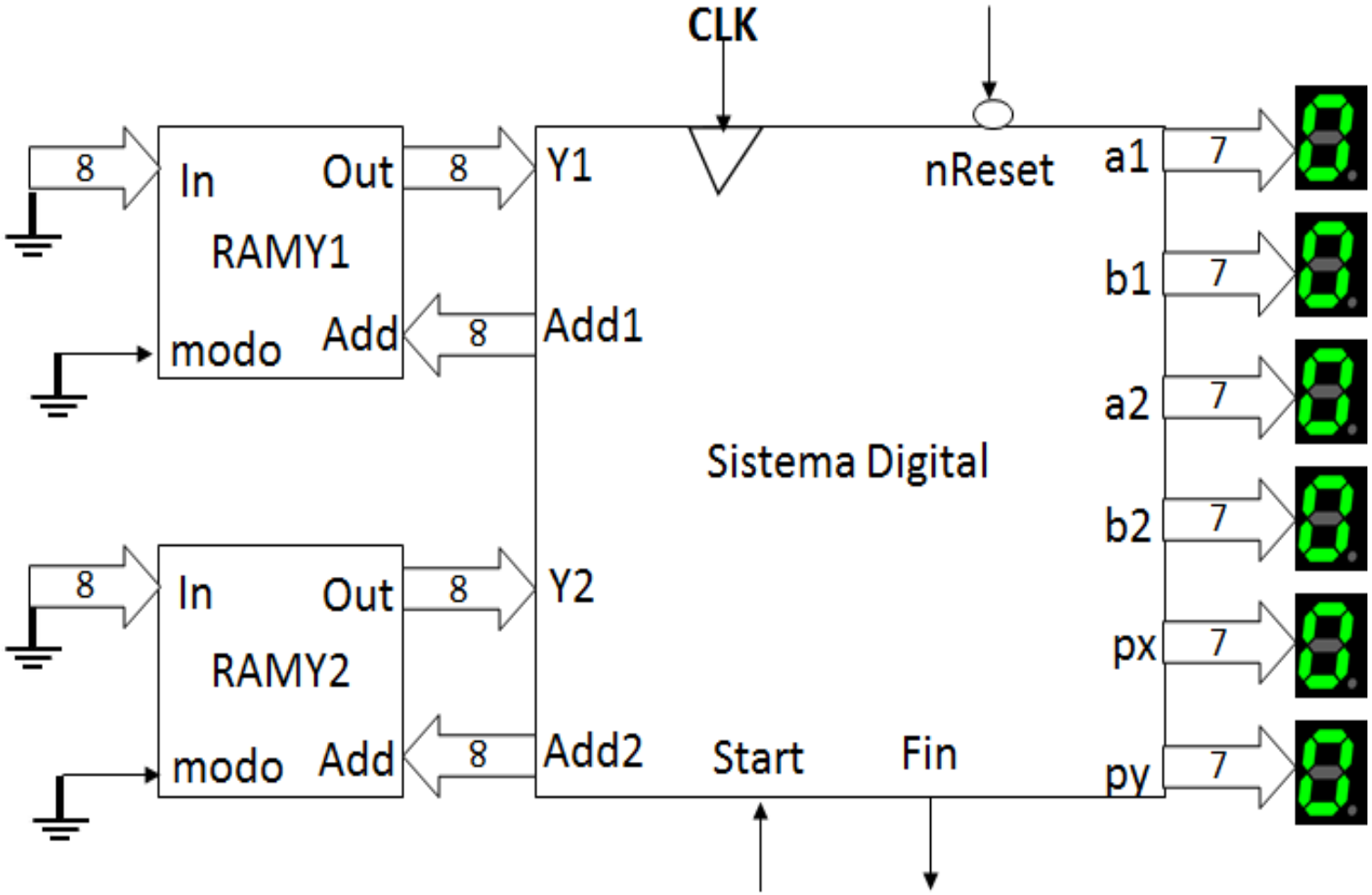
SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

CONJUNTOS

4.) Diseñar un sistema digital que lee dos RAM (RAMY1 y RAMY2) llenas con 255 puntos almacenados en memoria, estos puntos representan líneas rectas que se intersectan en algún punto. A medida que el sistema digital lee los puntos deberá encontrar las constantes a1, a2, b1, b2, px, py.

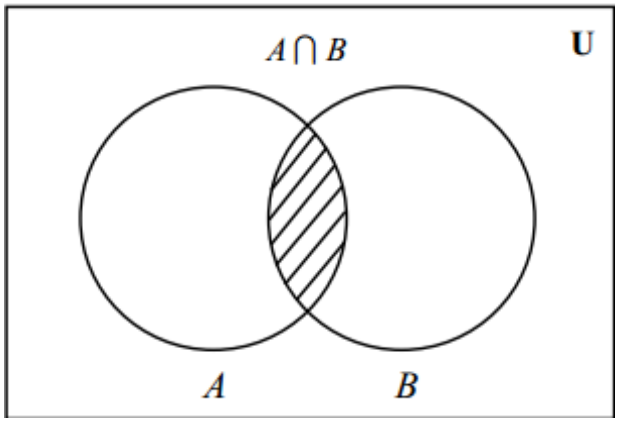




SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

5.) Diseñar un **SISTEMA DIGITAL DETECTOR DE INTERSECCIÓN DE CONJUNTOS**, el mismo permite el ingreso de dos conjuntos A y B en forma secuencial (byte a byte), luego detecte los números en común entre los conjuntos A y B ingresados.



- U: Conjunto de números de 8 bits (0 - 255).
- A: Conjunto de números de 8 bits, con un número de elementos ≤ 255.
- B: Conjunto de números de 8 bits, con un número de elementos ≤ 255.
- A∩B: Números en común de 8 bits delos conjuntos A y B.

SEÑALES:

- **InNúmero.-** Es una señal de 8 bits que el usuario usa para ingresar números uno a uno.
- **IngresoA.-** Señal que permite al usuario hacer el ingreso de número a número como elemento del conjunto A. (Asuma que el número está presente en la entrada InNúmero).
- **IngresoB.-** Señal que permite al usuario hacer el ingreso de número a número como elemento del conjunto B. (Asuma que el número está presente en la entrada InNúmero).

- **LedIngreso.-** Led indicador que todavía el usuario puede ingresar números. Este led puede apagarse debido a que el usuario presiona la tecla **FinIngreso** o debido a que el usuario ingresó el número máximo de elementos en cada conjunto (A, B).
- **Fin Ingreso.-** Esta señal que el usuario puede usar para indicar a la MSS que ya no desea ingresar más elementos en los conjuntos A y B.
- **Start.-** Luego de ingresar correctamente los elementos en cada conjunto el usuario deberá presionar la tecla **Start** para empezar a trabajar.
- **OK.-** Este led indica que la MSS terminó de encontrar los elementos de la intersección entre los conjuntos A y B.
- **A∩B.-** Esta señal de 8 bits de salida muestra número a número con un intervalo de 1segundo cada uno de los elementos de la intersección entre los conjuntos A y B. Luego que el sistema termina de mostrar todos los elementos de la intersección, la MSS regresa al estado inicial.

PRESENTAR:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

RECURSOS: https://goo.gl/lhtWFn

NOTA:

El pseudocódigo para el ingreso de los datos es el siguiente:

```
i=0;
j=0;
While(FinIngreso=0){
        LedIngreso=1;
        If (IngresoA=1 and i≤255){
                RAMA(i)=InNumero;
                i=i+1;
        }
        If (IngresoB=1 and j≤255){
                RAMB(j)=InNumero;
                j=j+1;
        }
        if(i=255 and j=255){break;}
}
```

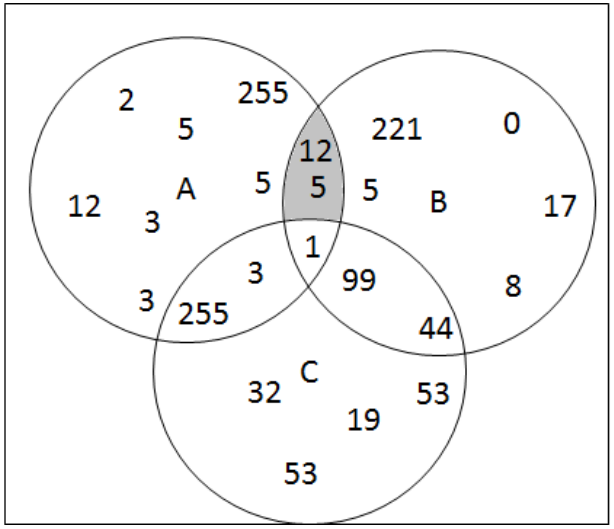
-- i representa el número de elementos en conjunto A, y j representa el número de elementos en B.

El pseudocódigo para la búsqueda de elementos es el siguiente:

```
#eA=i;
#eB=j;
i=0;
k=0;
While(i ≤ #eA){
    j=0;
    while(j ≤ #eB){
        if(RAMA(i)=RAMB(j){
            RAMA∩B(k)<=RAMA(i);
            k=k+1;
        }
        j=j+1;
    }
    i=i+1;
}
```

6.) Diseñe un sistema digital que recibe tres memorias RAM con 255 números (8 bits de direcciones), cada memoria RAM puede tener números repetidos almacenados. El sistema digital deberá identificar los números repetidos entre las memorias RAM_A y RAM_B, pero que no formen parte de la memoria RAM_C. Dichos números encontrados deberán ser mostrados uno a uno en tres display ya que los números almacenados pueden ir desde 0 a 255.

NOTA: de los números repetidos solo se tomará solo uno para encontrar la intersección. Además, deberá agregar las señales que crea conveniente para el funcionamiento del sistema completo.



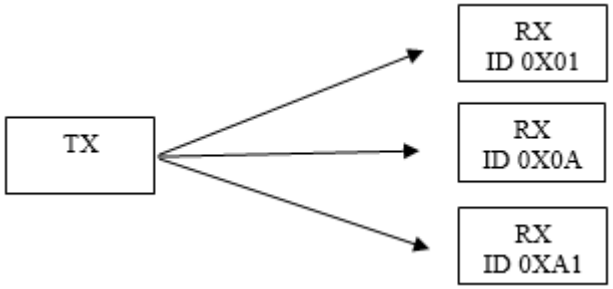
SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

COMUNICACIÓN SERIAL

7.) Realizar el diseño de un **SISTEMA DE COMUNICACIÓN CON VALIDACIÓN CHECKSUM**. El sistema está conformado por un transmisor que envía datos bajo un protocolo establecido hacia módulos receptores que validarán la trama recibida.

Protocolo de comunicación:			
Byte de Inicio	Byte ID de Equipo	Byte de Data	Byte Checksum
0X24	0X01 ó 0X0A ó XA1	0X00 – 0XFF	XOR



SEÑALES TX:

- **PaqueteOut.-** esta señal de 8 bits envía cada uno de los paquetes de la trama byte a byte, es decir envía de forma ordenada primero Byte de Inicio, Byte de ID, Byte de data y finalmente Byte de CheckSum.
- **SincTx.-** Esta señal sirve de sincronía para indicar al equipo receptor que existe un dato listo para ser leído.
- **StartTx.-** Señal para empezar a enviar los datos.
- **StopTx.-** Señal para detener el envío de datos.
- **IDRx.-** El Byte de ID será leído constantemente antes de enviar los datos, de esa forma puede enviar la trama a diferentes equipos.

SEÑALES RX:

- **PaqueteIn.-** esta señal de 8 bits recibe cada uno de los paquetes de la trama byte a byte, es decir recibe de forma ordenada primero Byte de Inicio, Byte de ID, Byte de data y finalmente Byte de CheckSum.
- **SincRx.-** Esta señal sirve de sincronía para indicar al equipo receptor que existe un dato listo para ser leído.
- **StartRx.-** Señal que empezar a recibir los datos.
- **StopTx.-** Señal para detener la recepción de datos.
- **IDRx.-** El byte de ID es cargado inicialmente antes de presionar start de tal forma que el ID del equipo de recepción se compara con el ID recibido.

PRESENTAR:

a) **Partición Funcional** del **Sistema Digital Completo (Tx y Rx)**.
b) **Diagrama ASM** de los circuitos **Controladores** del **Sistema Digital Tx y Rx** respectivamente, indicando claramente todas las salidas que deben ser generadas.
c) Código **VHDL** del sistema completo Tx en Quartus.
d) Código **VHDL** del sistema completo Rx en Quartus.

RECURSOS: https://goo.gl/X3fYLL

MATRICES

8.) Realizar el diseño de un **SISTEMA CALCULO DE DETERMINANTE DE UNA MATRIZ**. El sistema recibe una matriz cuadrada A(3x3) y luego procede a calcular el determinante de dicha matriz.

Ejemplo:
A=[1 5 8; 7 1 3;0 1 2]
B=det(A)

```
>> A=[1 5 8; 7 1 3;0 1 2]

A =

     1     5     8
     7     1     3
     0     1     2
```

```
>> B=det(A)

B = -15
```

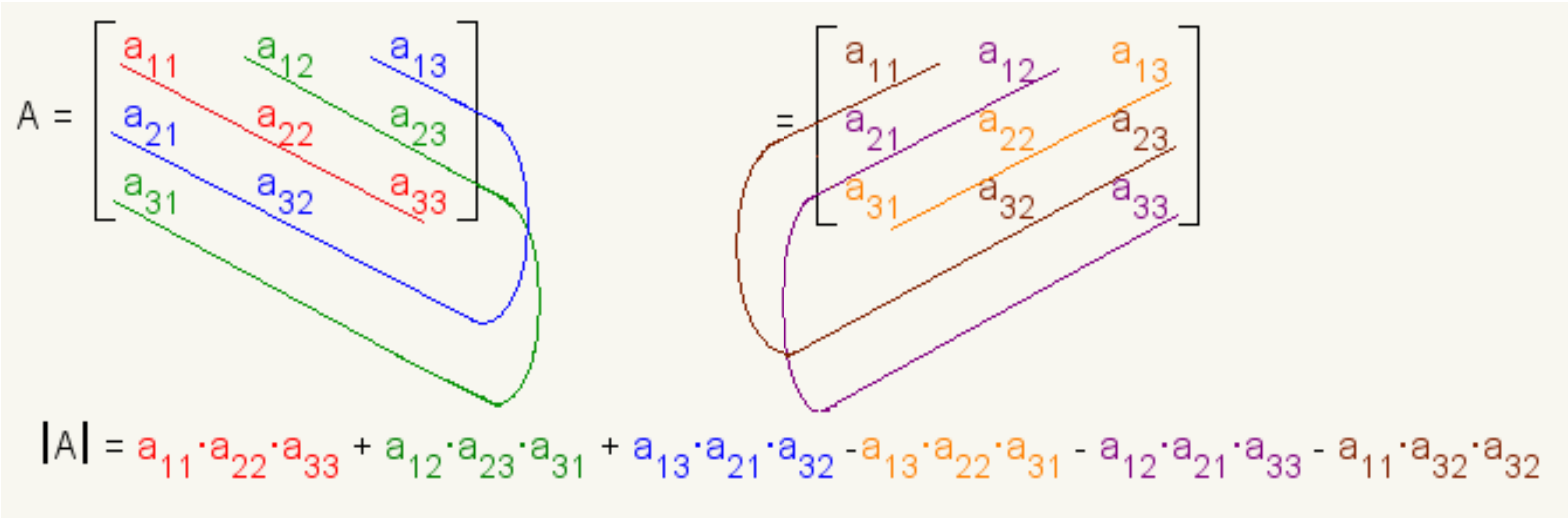
SEÑALES:

- **DatoA.-** esta señal de 4 bits permite el ingreso de cada uno de los números (0 - 9) de la matiz cuadrada A.
- **IngresoA.-** Esta señal indica cada vez que hay un número a ser ingresado en la matriz A.
- **Start.-** Da inicio al proceso del determinante luego de ser ingresada la matriz A.
- **LedOK.-** Led indicador de salida que da a conocer que el cálculo del determinante fue realizado con éxito.
- **DetA.-** Señal de salida de 8 bits que muestra el resultado del determinante de A.
- **Teclado**: permite el ingreso de los número entre 0-9.

PRESENTAR:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

NOTA: Para una matriz de 3x3 se pueden usar **Registros de Sostenimiento**. Pero se considerará puntos extras si se usan memorias RAM y si las matrices son de dimensiones mayores o iguales a 4 (nxn).



RECURSOS: https://goo.gl/IegCZI

9.) Realizar el diseño de un **SISTEMA MULTIPLICADOR DE MATRICES 3X4 Y 4X3**. El sistema recibe dos matrices, una A (3x4) y otra B (4x3) y luego procede a calcular la matriz C (3x3) que representa la multiplicación entre las matrices A y B.

Ejemplo:

Matriz A:			
3	0	2	3
0	9	1	0
5	1	3	4

.

Matriz B:		
1	2	2
9	0	0
8	1	1
6	2	4

=

Matriz C:		
37	14	20
89	1	1
62	21	29

SEÑALES:

- **DatoIn.-** esta señal de 4 bits permite el ingreso de cada uno de los números (0 - 9) de las matices A o B.
- **IngresoA.-** Esta señal indica cada vez que hay un número a ser ingresado en A.
- **IngresoB.-** Esta señal indica cada vez que hay un número a ser ingresado en B.
- **Start.-** Da inicio al proceso de multiplicación de las matrices luego de ser ingresadas.
- **LedOK.-** Led indicador de salida que da a conocer que la multiplicación fue realizada con éxito.
- **NumeroC.-** Señal de salida de 9 bits que muestra cada uno de los 9 números de 3 dígitos de la Matriz C.
- **Teclado**: permite el ingreso de los número entre 0-9.

PRESENTAR:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

NOTA: Se usan 9 bits para mostrar cada uno de los números, ya que en el peor de los casos cada número será 324 (1 0100 0100).

RECURSOS: https://goo.gl/oepRLw

NOTA: *Se puede usar el bloque MSI de multiplicación desarrollado en clase, pero recordar que consume ciclos de reloj en función de la cantidad de bits a multiplicar en la operación. El pseudocódigo para la multiplicación entre matrices es el siguiente.*

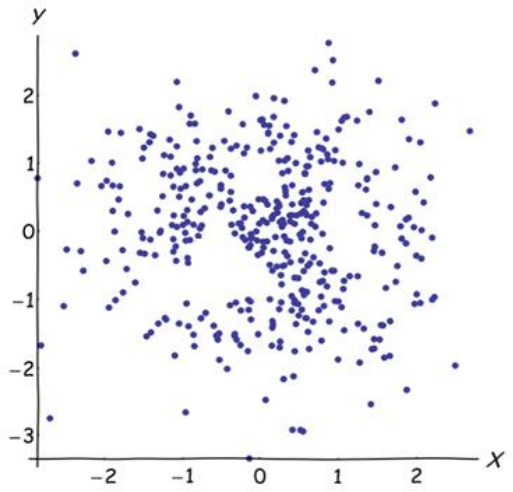
*Usaremos un circuito o función combinatorial **f()** que dado dos coordenadas de filas y columnas de una matriz nos da la coordenada vectorial en la memoria RAM:*

```
RAMC<={0 0 0; 0 0 0; 0 0 0};
for(i=0 to 3){
    for(k=0 to 3){
        for(j=0 to 4){
            RAMC(f(i,k))<=RAMC(f(i,k))+RAMA(f(i,j))*RAMB(f(j,k));
        }
    }
}
```

Matriz A (ixj):			
0:3	1:0	2:2	3:3
4:0	5:9	6:1	7:0
8:5	9:1	10:3	11:4

Matriz B (jxk):		
0:1	1:2	2:2
3:9	4:0	5:0
6:8	7:1	8:1
9:6	10:2	11:4

10.) En la siguiente gráfica se representa una nube de puntos (Máximo 20 Pts) de dos dimensiones, cada punto tiene un dato de 8bits que representa las coordenadas en **X** (4 bits menos significativos [0-3]) y en **Y** (4 bits más significativos [4-7]). Todos estos puntos deberán ser procesados por un Sistema Digital para crear una matriz de distancias las misma que estará almacenada en una **RAM_Distancia** de 20x20 direcciones. Para este objetivo deberá considerar las siguientes especificaciones:



RAM_Pts (20 datos)	
Address Pts	Coordenada del Punto 0x(y,x)
0x00	0x5F
0x01	0x1A
0x02	0X15
0x03	0X24
…	…
0x14	0XAA

i. Para calcular la distancia entre los puntos usaremos la técnica de la distancia Euclidiana, donde **P** representa las dimensiones en nuestro caso 2.

$$d_{ij} = \sqrt{\sum_{k=1}^{P} (x_{ik} - x_{jk})^2}$$

ii. Procedemos a calcular todas las distancias entre todos los pares de puntos (i,j), estos resultados estarán contenidos en la matriz como se indica en la figura siguiente. Una matriz de distancias es una matriz cuadrada, simétrica y con la diagonal principal con ceros. En nuestro caso la matriz es de 20x20 = 400 (Addres 9 bits).

Matriz de Distancia						
Pts.	A	B	C	D	…	S
A	0	2	10	22	…	4
B	2	0	6	8	…	8
C	10	6	0	15	…	11
D	22	8	15	0	…	25
…	…	…	…	…	0	26
S	4	8	11	25	26	0

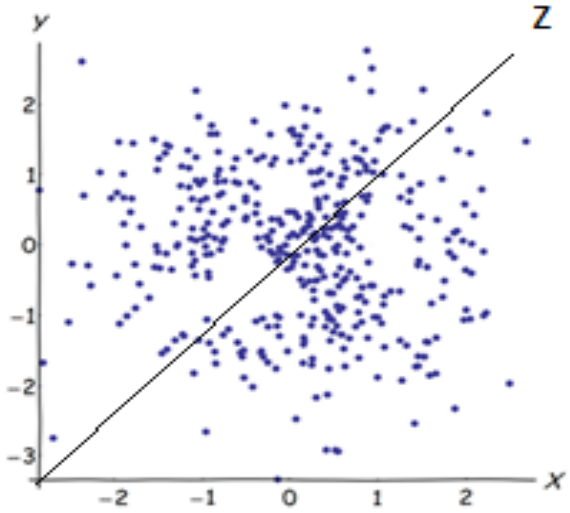
RAM_Distancia (400 datos)	
Address	Distancia entre dos puntos
0x00	0x00
0x01	0x02
0x02	0x0A
0x03	0x16
…	…
0x190	0x00

SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

11.) En la siguiente gráfica se representa una nube de puntos (Máximo 20 Pts) de tres dimensiones, cada punto tiene un dato de 9bits que representa las coordenadas en **X** (3 bits menos significativos [0-2]), en **Y** (3 bits siguientes [3-5]) y en **Z** (3 bits más significativos [6-8]). Todos estos puntos deberán ser procesados por un Sistema Digital para crear una matriz de distancias las misma que estará almacenada en una **RAM_Distancia** de 20x20 direcciones. Para este objetivo deberá considerar las siguientes especificaciones:

RAM_Pts (20 datos)	
Address Pts	Coordenada del Punto 0x(z,y,x)
0x00	0x5F
0x01	0x1A
0x02	0X15
0x03	0X24
…	…
0x14	0XAA



i. Para calcular la distancia entre los puntos usaremos la técnica de la distancia Euclidiana, donde **P** representa las dimensiones en nuestro caso 3.

$$d_{(p1,p2)} = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2}$$

01100001011100110110000101101110011110100110000 1

ii. Procedemos a calcular todas las distancias entre todos los pares de puntos (i,j), estos resultados estarán contenidos en la matriz como se indica en la figura siguiente. Una matriz de distancias es una matriz cuadrada, simétrica y con la diagonal principal con ceros. En nuestro caso la matriz es de 20x20 = 400 (Addres 9 bits).

Matriz de Distancia						
Pts.	A	B	C	D	…	S
A	0	2	10	22	…	4
B	2	0	6	8	…	8
C	10	6	0	15	…	11
D	22	8	15	0	…	25
…	…	…	…	…	0	26
S	4	8	11	25	26	0

RAM_Distancia (400 datos)	
Address	Distancia entre dos puntos
0x00	0x00
0x01	0x02
0x02	0x0A
0x03	0x16
…	…
0x190	0x00

SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

12.) Se tiene una matriz de distancias de 20 puntos, cuyos valores de distancia no son mayores a 22. Una Matriz de distancia es una matriz cuadrada, simétrica y con la diagonal principal con ceros, esta matriz de distancias estará almacenada en una **RAM_Distancia**, en nuestro caso con 400 direcciones (Addres 9 bits), la misma que deberá ser procesada con un Sistema Digital que construirá un dendrograma para agrupar los puntos en función de la distancia que poseen. Para este objetivo deberá considerar las siguientes especificaciones:

Matriz de Distancia						
Pts.	A	B	C	D	…	S
A	0	2	10	22	…	4
B	2	0	6	8	…	8
C	10	6	0	15	…	11
D	22	8	15	0	…	25
…	…	…	…	…	0	26
S	4	8	11	25	26	0

RAM_Distancia (400 datos)	
Address	Distancia entre dos puntos
0x00	0x00
0x01	0x02
0x02	0x0A
0x03	0x16
…	…
0x190	0x00

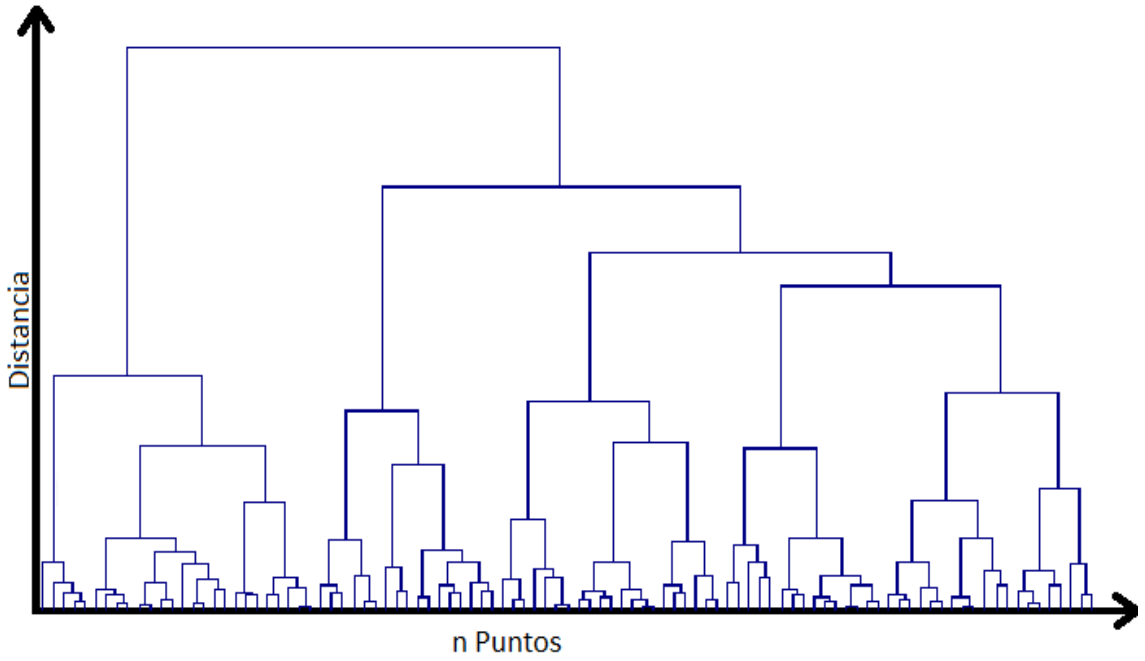
i. La primera agrupación de puntos se hace en base a la matriz de distancias. Luego se agrupan los dos puntos diferentes de distancia menor.

ii. A continuación, tenemos que comparar la distancia de un punto con otros puntos y a su vez con el grupo de puntos anterior. Por lo tanto, para calcular la distancia entre un punto y un grupo o la distancia entre dos grupos, usaremos el criterio de la distancia mínima. Dist((a,b….,i), x) = min{Dist(a,x), Dist(b,x),……, Dist(i,x)}

NOTA: asuma que la memoria RAM_Distancias ya está llena con las 400 distancias.

SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.



OPERACIONES DIGITALES

13.) Realizar el diseño de un **SISTEMA QUE RESUELVE MAPAS DE KARNAUGH (MK)**. El sistema recibe un mapa de 3 variables y luego procede a calcular el número de agrupaciones **(n)** más eficientes de dicho MK.

Ejemplo:

y0 \y2, y1	00	01	11	10
0	0: 1	2: 1	6: 1	4: 1
1	1: 1	3: 1	7: 1	5: 1

n=1

y0 \y2, y1	00	01	11	10
0	1	1	1	0
1	0	1	1	1

n=3

SEÑALES:

- **DatoMK.-** esta señal de 8 bits que representa todo el conjunto de unos y ceros presentes en el MK.
- **IngresoMK.-** Esta señal indica cuando se quiere ingresar el único dato de 8 bits.
- **Start.-** Da inicio al proceso de resolución del MK.
- **MOK.-** Led indicador de salida que da a conocer que la resolución fue realizada con éxito.
- **#eMK.-** Señal de salida de 4 bits que muestra el mejor número de elementos encontrados en la resolución del MK.

PRESENTAR:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

NOTA: se considerará puntos extras si el MK pueden ser de un número de variables mayor a 4 (y3,y2,y1,y0).

RECURSOS: https://goo.gl/rXOnLl

14.) Diseñar un Sistema Digital que resuelve diagramas de estados que dada una RAM de 8 direcciones donde se escriben operadores identificados por un número ID como se indica en la siguiente tabla:

ID Operador	Operador
0x01	and(y2, y1, A)
0x02	nand(y2, y1, A)
0x03	and(nand(y2, y1), A)
0x04	and(y2, nand(y1, A))
0x00	Termino sin Operador

El sistema deberá evaluar la RAM de 8 direcciones, donde cada 2 direcciones se representan dos bits del decodificador de estado siguiente y el decodificador de salida, como si indica a continuación:

Decoder	Dir RAM	Dato Ram
Y1	0x00	0x01
	0x01	0x03
Y2	0x02	0x02
	0x03	0x04
Out	0x04	0x03
	0x05	0x01
-	0x06	0x00
-	0x07	0x00

El sistema deberá retornar una memoria RAM que represente la tabla de estados presentes y siguientes tal como se indica en la siguiente tabla:

Dir RAM	Dato RAM (y2,y1,A,Y1,Y2,Out,x,x)
0x00	%000 010 --
0x01	%001 110 --
0x02	%010 010 --
0x03	%011 110 --
0x04	%100 011 --
0x05	%101 111 --
0x06	%110 011 --
0x07	%111 101 --

NOTA: la memoria RAM de salida deberá ser inicialmente puesta en cero.

SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

15.) Diseñar un Sistema Digital que simplifica ecuaciones booleanas, para lo cual deberá recibir una RAM de 8 direcciones donde se escriben operadores identificados por un número ID como se indica en la siguiente tabla:

ID Operador	Operador
0x00	Termino sin Operador
0x01	and(not(y2), not(y1), y0)
0x02	and(not(y2), y1, not(y0))
0x03	Termino sin Operador
0x04	Termino sin Operador
0x05	and(y2, not(y1), y0)
0x06	and(y2, y1, not(y0))
0x07	and(y2, y1, y0)

El sistema deberá evaluar la RAM de 8 direcciones, donde todas cada dirección tendrá el valor de una de las operaciones (0x01 – 0x07) las que tengan el valor (0x00) no deberán ser evaluadas, además NO deberán repetirse operaciones en la RAM, como si indica a continuación:

Variable	Dir RAM	Dato RAM (término de la ecuación)
Salida	0x00	0x01
	0x01	0x05
	0x02	0x06
	0x03	0x02
	0x04	0x07
	0x05	0x00
	0x06	0x00
	0x07	0x00

El sistema deberá retornar una memoria RAM que represente los términos de la ecuación simplificada:

ID Operador	Operador
0x00	Término sin Operador
0x01	and(y1, not(yo))
0x02	and(y2, y1)
0x03	and(y2, y0)
0x04	and(not(y1), y0)
0x05	and(not(y2), y1, not(y0))
0x06	and(not(y2), not(y1), y0)
0x07	and(y2, y1, y0)
0x08	and(y2, y1, not(y0))
0x09	and(y2, not(y1, y0))

En las direcciones de memoria donde no hay términos se los llena con 0x00.

Dir RAM	Dato RAM (y2,y1,y0,Salida,x,x,x,x)
0x00	0x01
0x01	0x02
0x02	0x03
0x03	0x04
0x04	0x00
0x05	0x00
0x06	0x00
0x07	0x00

NOTA: la memoria RAM de salida deberá ser inicialmente puesta en cero.

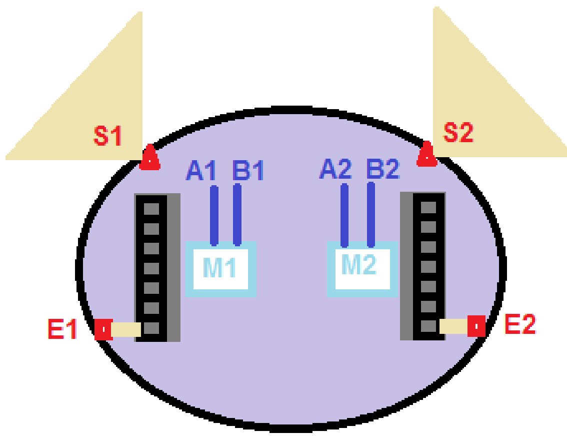
SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

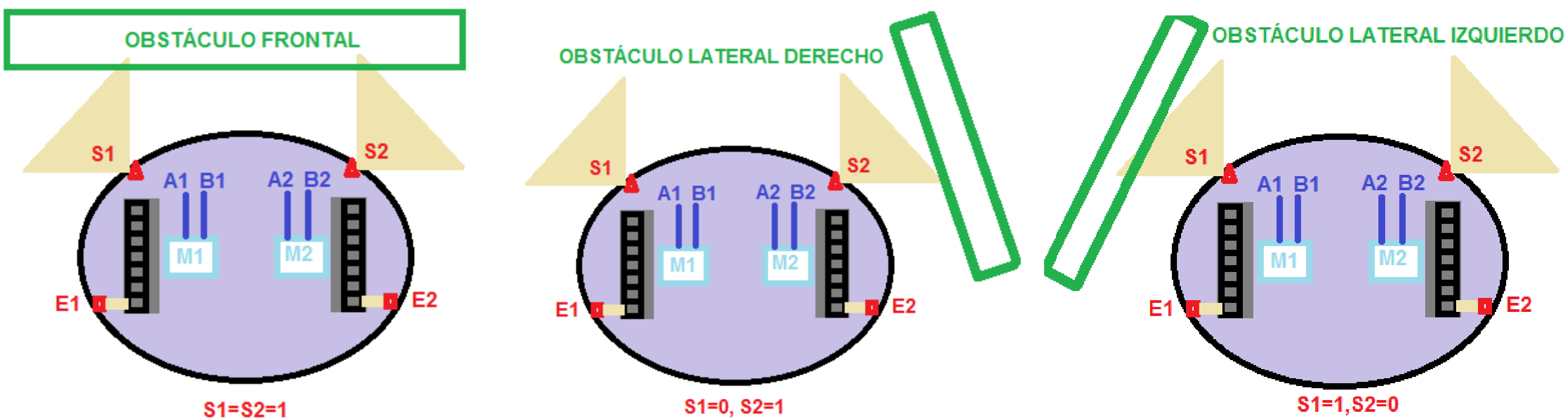
APLICACIONES

16.) Diseñar un sistema digital que gobierna un **DOMOBOT (Robot de uso doméstico)**. Para dar inicio se debe presiona y soltar el botón de **START**.

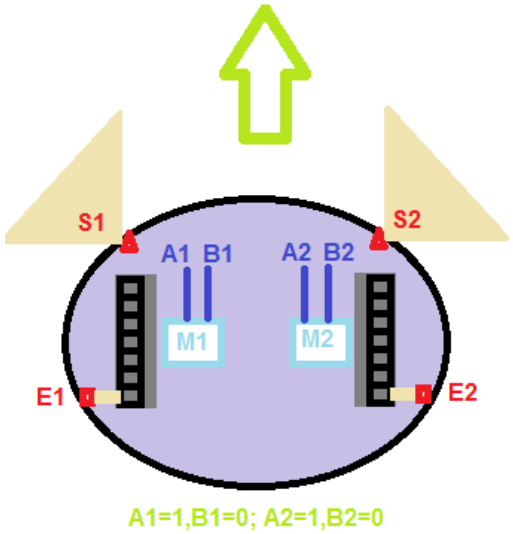
S1, S2: Sensor de distancia.
E1, E2: Encoder optico.
A1, B1: Control de Motor 1.
A2, B2: Control de Motoro 2.

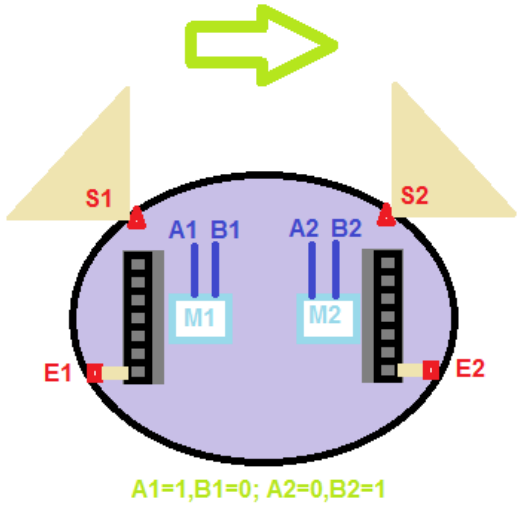


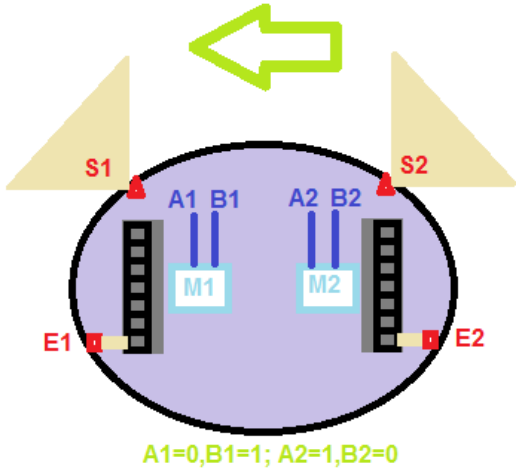
El Domobot cuenta con dos sensores (S1 y S2) (botoneras) que le permiten detectar cuando está cerca de una superficie, según las combinaciones que se indican a continuación.

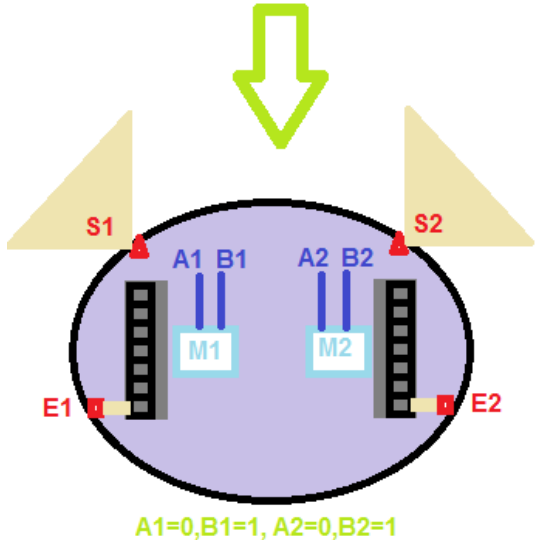


El domobot utiliza una tracción diferencial para desplazarse, la combinación de las señales A y B (leds) deberán ser en función de hacia que dirección queremos que el robot se dirija.









0110000101110011011000010110111001111010011000011

El sensor E1 y E2 (botoneras) detectan cuantos grados avanza cada rueda al momento de desplazar el robot, sabiendo que cuando una rueda da una vuelta completa el sensor **E** habrá dado 4 pulsos y el robot habrá avanzado 10cm.

Requerimientos:

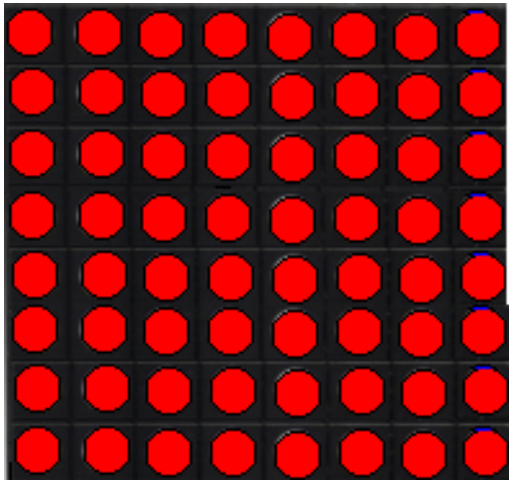
El robot deberá ser capaz de evadir obstáculos, además está programado para hacer un desplazamiento de 40cm (con o sin obstáculos) y luego regresar al punto inicial. El movimiento de evasión de obstáculos será:

1. Obstáculo Frontal -> Girar 180°
2. Obstáculo lateral derecho -> Girar 90° a la izquierda.
3. Obstáculo lateral izquierdo -> Girar 90° a la derecha.

Presentar:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

17.) Realizar el diseño de un Sistema Digital **MARQUESINA CON MATRIZ DE LEDS**. El sistema tiene como interfaz de salida cuatro **matrices** de leds de 8x8 que permite visualizar letras que estarán pregrabadas en una memoria RAM de 8 direcciones, las letras deberán visualizarse mientras se desplazan de izquierda a derecha. El ingreso de las letras a la memoria RAM se lo hará por medio de un teclado alfabético, para lo cual se deberá presionar la tecla Grabar letra por letra y luego de presionar la tecla Start. Si luego de grabar el mensaje el usuario presiona la tecla Mostar, el sistema empezará a hacer aparecer las letras en la marquesina, letra a letras empezará a aparecer hasta que se vea una cuatro letras en la marquesina a la vez. Una vez que se pierde la última letra del mensaje de 8 caracteres, el sistema mostrará nuevamente el mensaje, todo esto hasta que el usuario presione la tecla salir.



SEÑALES:

- **Tecla Start.-** el sistema permitirá grabar mensaje o mostrar mensaje siempre y cuando se presione Start primero.
- **Tecla Reset**: Esta tecla solo deberá resetear a la MSS, es decir la MSS será quien resetee los bloques MSI que sean necesarios.
- **Tecla Grabar.-** Permitirá ingresar letra a letra el mensaje a ser mostrado, el mensaje tiene máximo 8 letras ya que serán almacenadas en una ram de 8bits x 8 direcciones.
- **Tecla Mostrar.-** Permitirá al sistema empezar a mostrar el mensaje mientras se desplaza.
- **Tecla Stop.-** Esta tecla permite detener la marquesina y si luego se presiona Grabar, se podrá ingresar un nuevo mensaje, caso contrario si se presiona mostrar se vuelve a mostrar el mensaje almacenado anteriormente.
- **Matriz de leds.-** matriz de 8x8 leds que permiten formar cualquier letra, pero para encender los puntos para las letras se deberán hacer multiplexaciones.
- **Teclo alfabético.-** Teclado con la letras del alfabeto para ingresar cualquier mensaje.

NOTA: la velocidad de desplazamiento del mensaje es 2puntos / segundo.

SE PIDE:

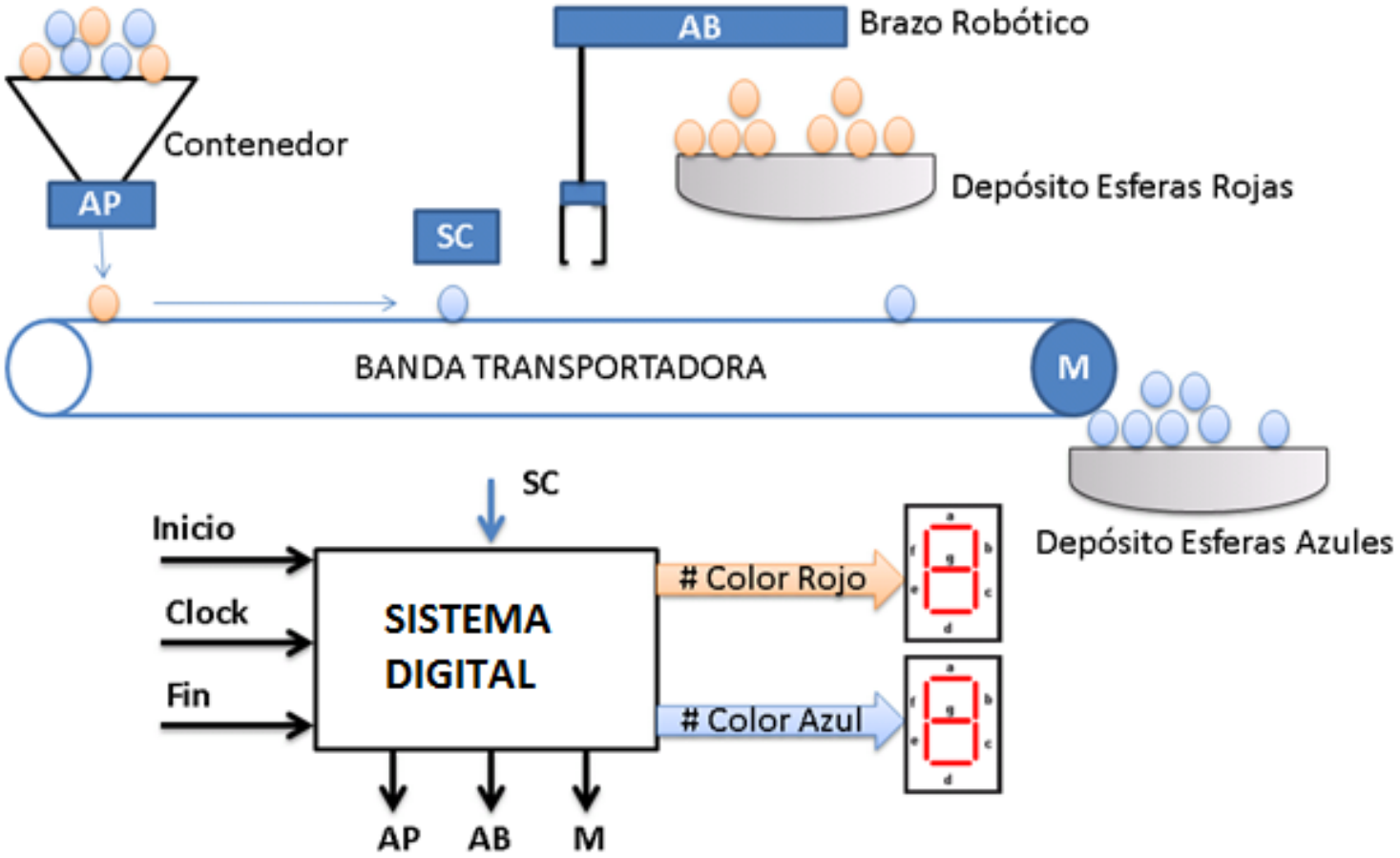
a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

18.) Se tienen un proceso de clasificación de esferas por color, esferas que inicialmente se encuentran almacenadas en un **contenedor** que el la salida de las esferas tiene acoplado un **Actuador de Paso (AP)** que permite pasar una sola esfera por vez. Luego cada esfera cae a la **banda transportadora** hasta llegar al **sensor de color (SC)** y si es la esfera roja un **actuador de brazo robótico (AB)** que levanta la lleva al **depósito de esferas rojas**, caso contrario avanza por la banda transportadora hasta llegar al **depósito de esferas azules**.

Se quiere realizar un sistema Digital que luego de presionar y soltar el botón **INICIO**, accione el **Motor (M)** de la **banda transportadora**, luego activará el **actuador de paso** para permitir que caiga una esfera por vez a la banda (la válvula deberá ser activada por 5 segundos por su respuesta mecánica). Una vez que una esfera está en la banda transportadora ésta pasará por el **sensor de color** el cual determinará el destino final de la esfera. Si la esfera es azul, la banda sigue trabajando hasta llevar a ésta al contenedor de esferas azules que queda al final de la banda transportadora. Si el color de la esfera es rojo, la banda transportadora deberá detenerse por 15 segundos que es el tiempo estimado que el brazo robótico demora en ***bajar - tomar a esfera - llevar la esfera - dejar la esfera - regresar a su posición original***. El **actuador de paso** se activará automáticamente si el sensor de color detecta que es una esfera azul, pero si es una esfera roja el actuador de paso se activará luego de que el brazo robótico llegue a su posición inicial. Adicionalmente el sistema permitirá ***visualizar en dos Display*** la cantidad de esferas rojas y azules en tiempo real. El sistema regresará a su estado inicial cuando el operador presiona el botón de **FIN** o cuando uno de los dos depósitos alcance la cantidad de nueve esferas.

SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.



19.) Realizar el diseño de un **SISTEMA DE ALARMA COMUNITARIA** para una ciudad, el cual notifica al UPC más cercana lo siguiente: el número cuadra, número de casa y la emergencia que tienen los habitantes de esa casa.
El sistema cuenta con un equipo transmisor en cada casa (máximo 99 casas por cuadra), la ciudad en total tiene máximo 90 cuadras, cada alarma de casa debe ser capaz de enviar al UPC una trama de comunicación de 5 bytes de datos, en el siguiente formato:

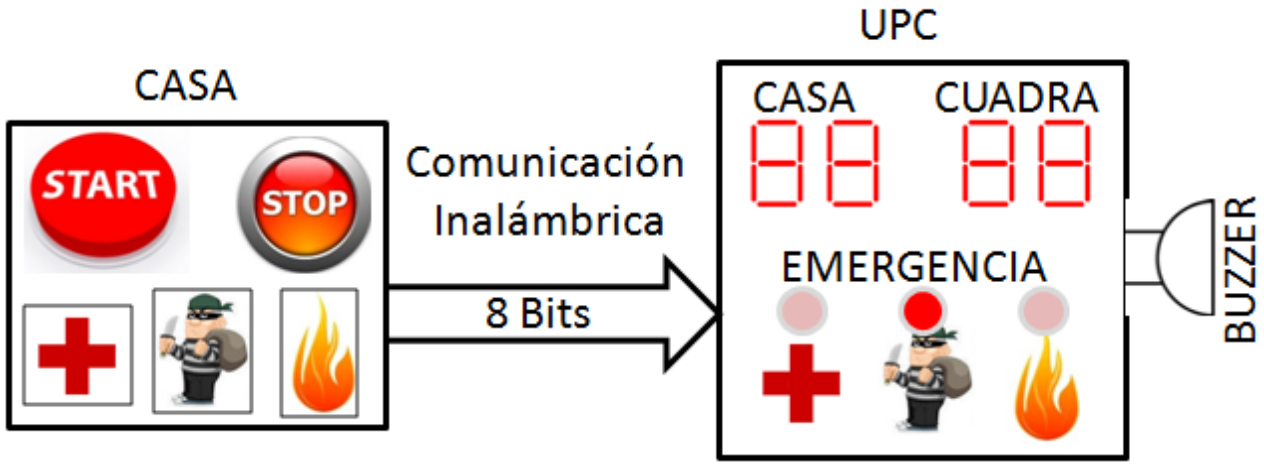
1: Byte de Inicio	2: Byte ID Cuadra	3: Byte ID Casa	4: Emergencia	5: Byte de Fin
0xF0	0x00 – 0x5A	0x00 – 0x63	0x01 – 0x03	0xF1

Los tipos de emergencia del sistema de alarma son:

1. Emergencia Médica
2. Robo
3. Incendio

El Sistema de monitoreo que está en la UPC tiene 4 displays de 7 segmentos, 3 leds para indicar la emergencia y una salida digital para un Buzzer como indicador acústico. Este sistema deberá validar los Bytes de inicio-Fin para luego mostrar la información a los policías permanentemente durante 30 segundos. En ese tiempo el Buzzer deberá sonar a una Frecuencia de 5kHz (Señal Cuadrada con un ciclo de trabajo del 50% generada por el sistema del UPC).
Ambos sistemas por separado tienen un pulsador de **START** y un botón de **RESET** para reiniciar cada sistema.

NOTA: *Asuma que la comunicación es inalámbrica con modulación QPSK y HalfDuplex, de Punto - Multipunto. Imagine que siempre hay línea de vista.*



SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

20.) Realizar el diseño de un sistema Digital de **CONTROL DE PERSONAL**. El sistema controla el ingreso y salida de forma automática del personal de trabajo (máximo 9). Cuando un trabajador entra o sale debe:

1. Con 4 botones indicará: **entrada trabajo, pausa inicio, pausa fin y salida de trabajo**.
2. Con un teclado decimal ingresará su número de **ID de trabajador** (1-9)
3. Usando el teclado decimal se autenticará con un **password** único por persona de 4 dígitos.

Dos botones para indicar que está saliendo o ingresando, un pulsador que le sirve para iniciar el sistema (**START**), un botón de **RESET** para inicializarlo todo. Al finalizar la jornada de trabajo (máximo 12 horas) el trabajador se autenticará y el sistema debe indicar en dos displays de 7 segmentos las horas de trabajo de cada persona (redondear hacia abajo) y además el ID del trabajador en otro display, información que el departamento de Talento Humano usará para hacer el pago por hora.

IMPORTANTE: el sistema debe permitir **REGISTRAR** los empleados inicialmente y cada uno con su clave de registro.

NOTA: *Asuma que el registro se lo hace una sola vez. Los trabajadores podrán pausar su trabajo para comer.*

SE PIDE:

a) Hacer la partición funcional del sistema completo: **Mss, Ram, Msi,** etc.
b) Diagrama **ASM** del controlador.
c) Código **VHDL** del sistema completo en Quartus.

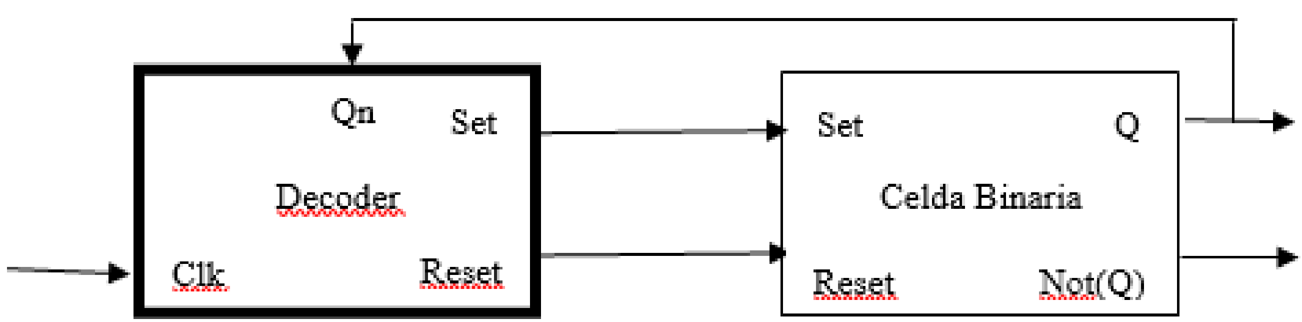
RECURSOS: https://goo.gl/0t6dUA

01101010010010100110000101101110

MSA

21.) Diseñar un Maquina Secuencial Asíncrona (MSA) que hace el trabajo de un decodificador que en conjunto con una celda Binaria trabaja como un **Flip-Flop Inv**.

Flip-Flop Inv				**Celda binaria**				
Clk	Qn	Qn+1		Set	Reset	Qn	Qn+1	
$\uparrow$	0	1	Inv	0	0	0	0	Hold
$\uparrow$	1	0	Inv	0	0	1	1	Hold
$\not\uparrow$	0	0	Hold	0	1	0	0	Rst
$\not\uparrow$	1	1	Hold	0	1	1	0	Rst
				1	0	0	1	Set
				1	0	1	1	Set
				1	1	0	$\emptyset$	$\emptyset$
				1	1	1	$\emptyset$	$\emptyset$

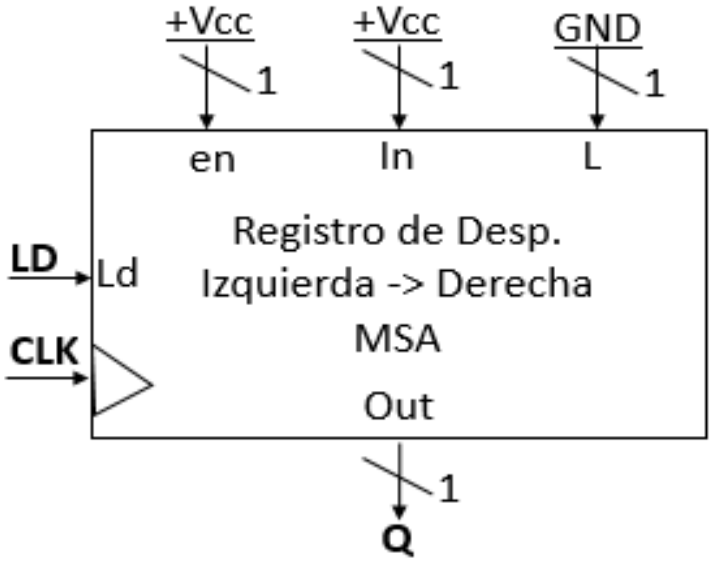


PRESENTAR:

a) Diagrama de estados primitivo.
b) Mapa de estados primitivo.
c) Diagrama de equivalencia máxima.
d) Diagrama de estados reducido.
e) Asignación de código de estados.
f) Mapa de excitación.
g) Decodificador de estado siguiente y decodificador de salida.
h) Implementar el circuito completo.

22.) Usando las técnicas de diseño asincrónico, diseñe un Registro de desplazamiento con entradas constantes. Tome en cuenta para el diseño solo las entradas **CLK** y **LD**, la salida **Q** es de un solo bit, de acuerdo al formato y tabla característica mostrados a continuación. Formato: **CLK, LD/Q**

CLK	LD	Qi	Qi+1	Función
↑	0	0	0	Desplazamiento
		1	0	
	1	0	1	Load
		1	1	
↑	0	0	0	Hold
		1	1	
	1	0	0	Hold
		1	1	



Presentar:

a) Diagrama de estados primitivos.
b) Mapa de estados primitivo.
c) Tabla de implicantes y Diagrama de equivalencia máxima.
d) Diagrama de estados reducido y asignación de códigos de estado.
e) Mapa de excitación.
f) Mapas del decodificador de estado siguiente y salida de la MSA.
g) Diagrama completo de la MSA.

01100001011100110110000101101110011110100110000 1