Práctica 1

Diseño VHDL de Sistemas Combinacionales.

Descripción del problema:

Se desea diseñar un circuito combinacional con 2 entradas de 2 bits cada una X_1X_0 e Y_1Y_0, y una única salida Z que toma valor '1' si el número X_1X_0 es menor que el número Y_1Y_0.

Empezaremos la práctica trasladando los datos que nos ha proporcionado el enunciado a una tabla de verdad para su posterior uso.

Tabla de verdad.

X_1	X_0	Y_1	Y_0	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Apartado A.

Obtener la expresión más simplificada de Z en forma de suma de productos. Describir en VHDL la expresión obtenida utilizando una asignación concurrente. El nombre de esta arquitectura debe ser concurrente_sdp.

Trasladaremos los datos de la tabla a un mapa de Karnaugh.

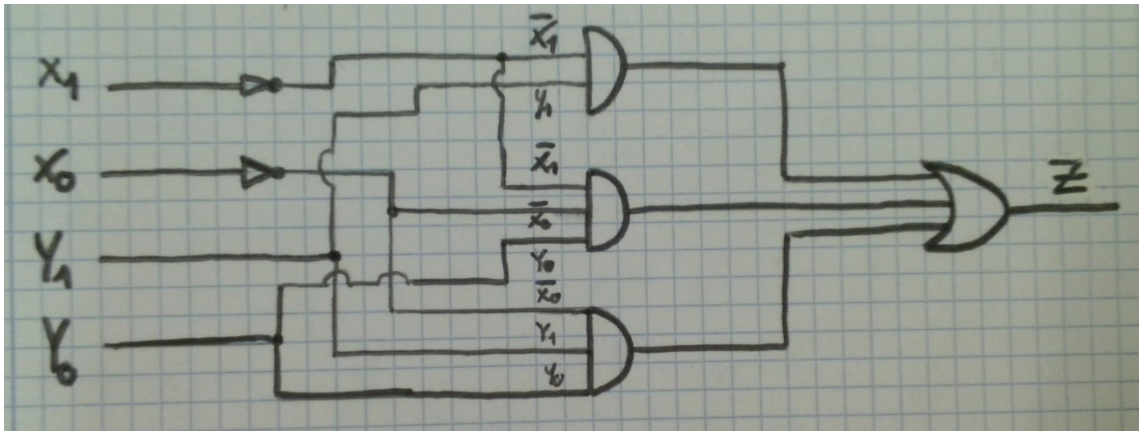
Al pedirnos la suma de productos, utilizaremos el mapa de Karnaugh para hallar la primera forma canónica para poder obtener una función simplificada de Z.

Z	Y_1=0, Y_0=0	Y_1=0, Y_0=1	Y_1=1, Y_0=1	Y_1=1, Y_0=0
X_1=0, X_0=0	0	1	1	1
X_1=0, X_0=1	0	0	1	1
X_1=1, X_0=1	0	0	0	0
X_1=1, X_0=0	0	0	1	0

Esto nos dará como resultado la siguiente expresión:

Z=*

Una vez hallada la ecuación, procederemos a dibujar la circuitería necesaria para realizarla.



Apartado B.

Obtener la expresión más simplificada de Z en forma de producto de sumas. Describir en VHDL la expresión obtenida utilizando una asignación concurrente. El nombre de esta arquitectura debe ser concurrente_pds.

Trasladaremos nuevamente los datos de la tabla a un mapa de Karnaugh.

Al pedirnos el producto de las sumas, utilizaremos el mapa de Karnaugh para hallar la segunda forma canónica para poder obtener una función simplificada de Z.

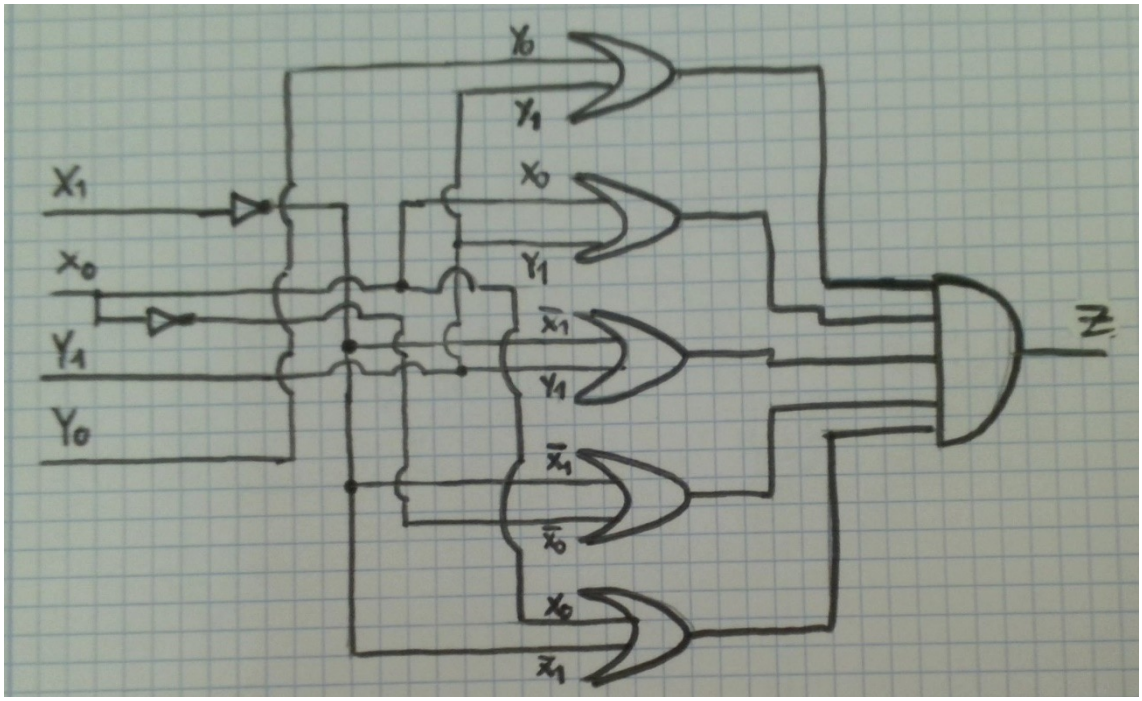
Z	Y_1=0, Y_0=0	Y_1=0, Y_0=1	Y_1=1, Y_0=1	Y_1=1, Y_0=0
X_1=0, X_0=0	0	1	1	1
X_1=0, X_0=1	0	0	1	1
X_1=1, X_0=1	0	0	0	0
X_1=1, X_0=0	0	0	1	0

Esto nos dará como resultado la siguiente expresión:

Z=(

Una vez hallada la ecuación, procederemos a dibujar la circuitería necesaria para realizarla.

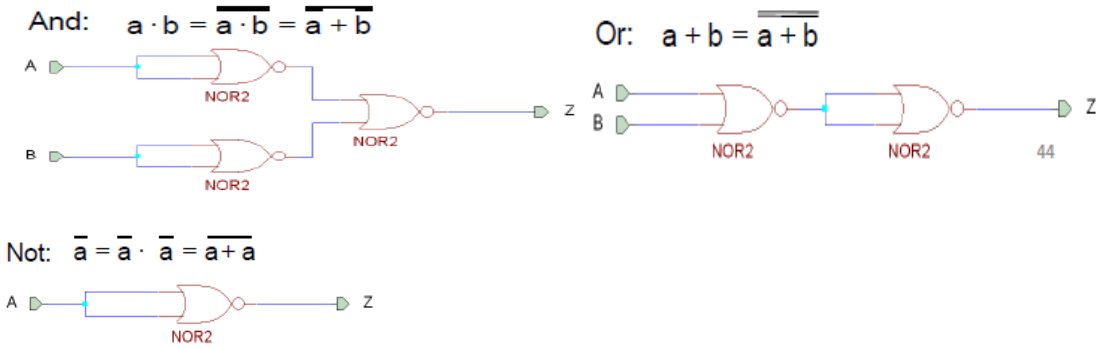
Cambiar la segunda puerta a ver si funciona.



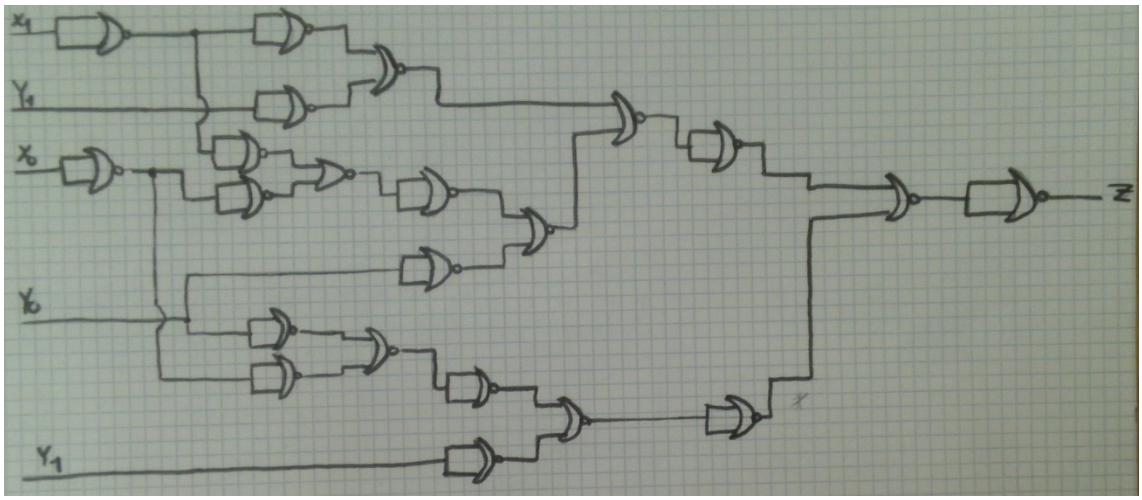
Apartado C.

Implementar Z utilizando sólo puertas NOR (y opcionalmente inversores).

Para realizar este apartado, nos hemos apoyado en los cambios equivalentes que se podrían realizar tal como veíamos en las diapositivas expuestas en clase. Es por ello que hemos utilizado los siguientes cambios:



Tras realizar estos cambios, nos da como resultado un circuito con esta forma:

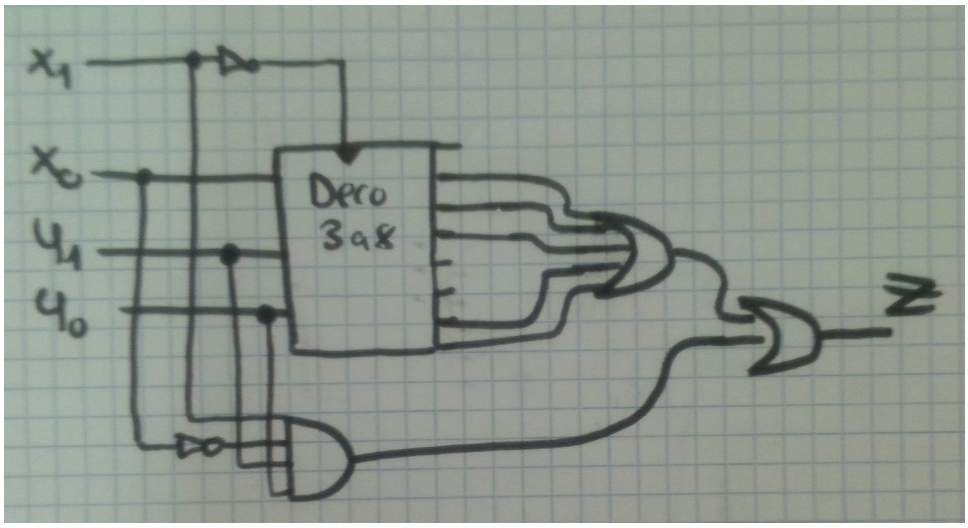


Apartado D.

Implementar Z utilizando un decodificador de 3 a 8 y puertas lógicas auxiliares

Utilizaremos un solo decodificador de 3 a 8, conectando a la entrada de Enable para cerciorarnos así de que sólo funciona cuando =0.Uniremos todas las salidas que hacen Z=1 a una puerta OR, para que en el caso de que una sea positiva, se de salida Z=1.

Como con el decodificador de 3 a 8 no nos es suficiente, debido a que queda un caso en el que se hace positiva la salida, es necesario implementarla mediante una puerta AND y más tarde unirla con una OR a las salidas del Decodificador.



Apartado e.

Implementar Z utilizando un multiplexor de 8 a 1 y puertas lógicas auxiliares

En base a la tabla de verdad hemos reducido las posibilidades a 8 casos y en función de esos caso, introduciremos 0 si en ambos son la salida 0, un 1 si en ambos caso la salida es 1 y lo introduciremos en los casos en que se hace 1 la salida y la entrada debería ser un valor de X=0.