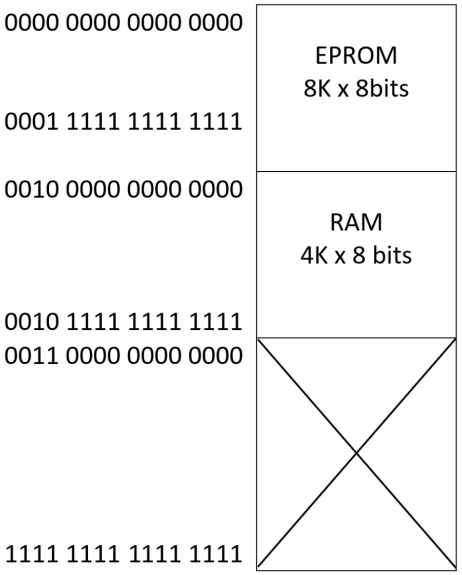
PRÁCTICO 4 - Direccionamiento y Lógica de Decodificación de Memorias

Ejercicio 4:

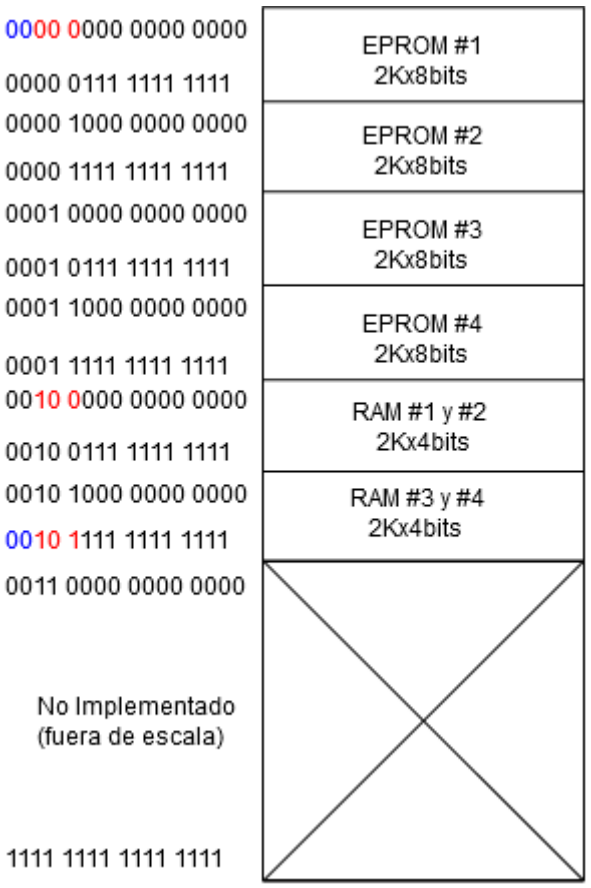
Construir un sistema de memoria como el que se muestra en el mapa de memoria de la figura. Se dispone para su implementación con los siguientes “chip” de memoria: EPROM de 2K x 8 bits y RAM de 2K x 4 bits.

A. Realizar una implementación que NO genere posiciones imagen en el espacio no implementado.
B. Realizar una implementación en la cual se generen posiciones imagen del contenido de la EPROM y la RAM a lo largo de todo el espacio direccionable. Analizar: ¿cuántas veces se replica el contenido de la RAM? y ¿cuántas veces se replica el contenido de la EPROM?, ¿por qué?



Respuesta A:

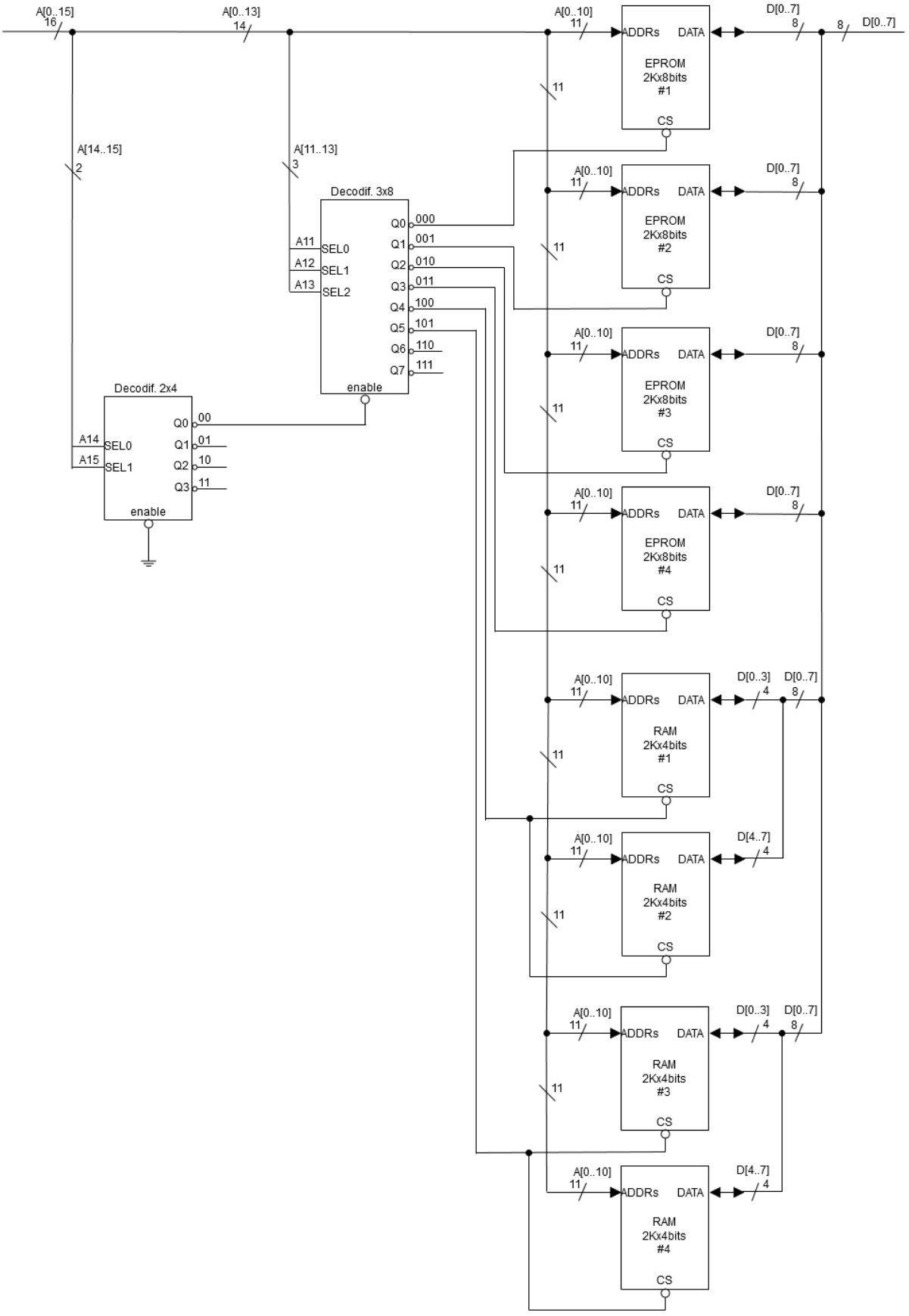
Para implementar la EPROM de 8K x 8bits se necesitan 4 chip de 2K x 8bits, conectados en serie. Para implementar la RAM de 4K x 8bits se requieren 2 chip de 2K x 4bits conectados en paralelo, para formar 2K x 8bits, y luego 2 de estos bloques conectados en serie. Como resultado, el mapa de memoria implementado tiene la siguiente forma:



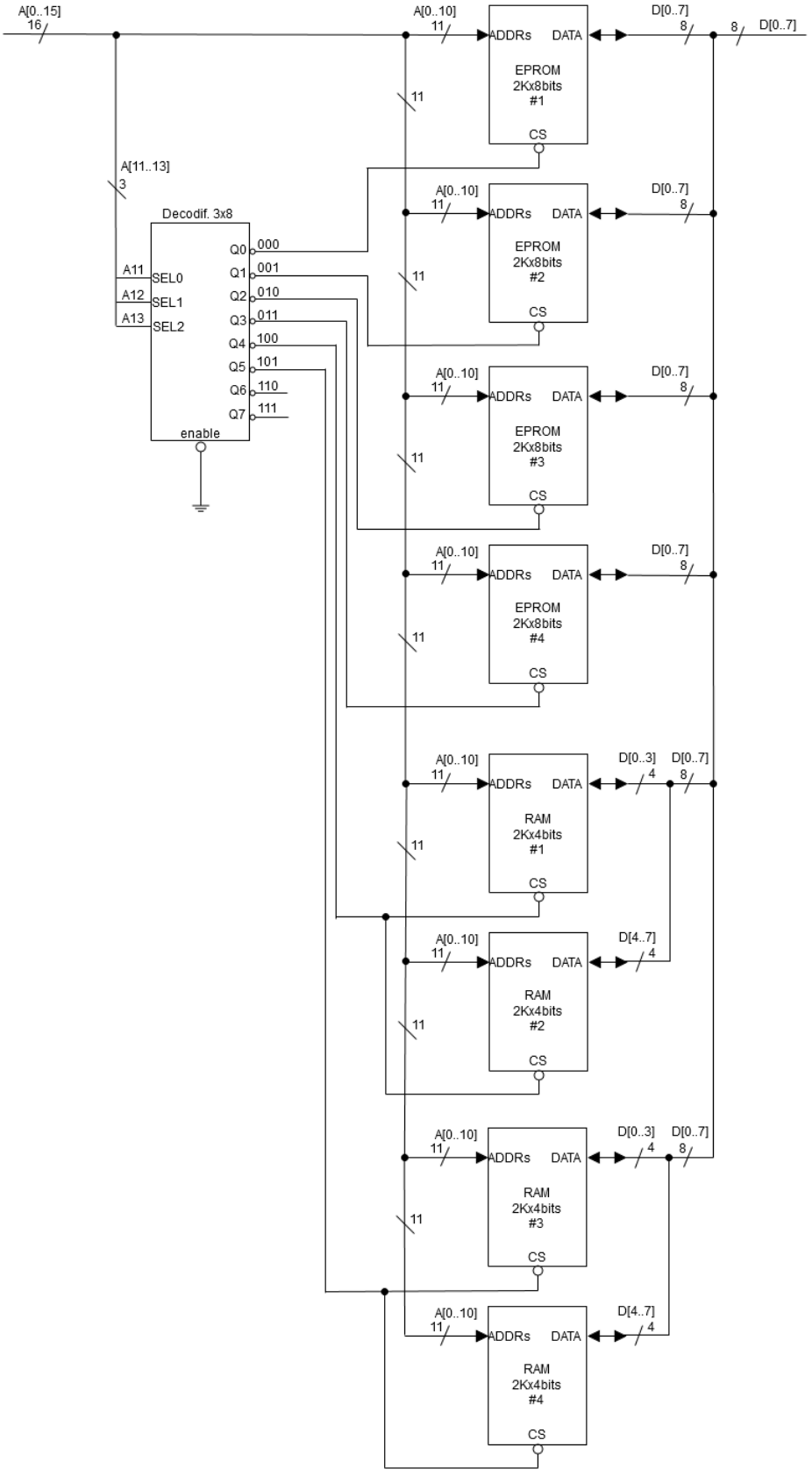
Analizando el mapa vemos:

- Para direccionar las 2K palabras dentro de cada bloque de memoria se necesitan 11 bits de address (A10..A0).
- Para direccionar las 8K palabras de la memoria EPROM se requieren 13 bits de address, por lo que se utilizan los bits A12 y A11 para seleccionar cada uno de los 4 chip.
- Para direccionar las 4K palabras de la memoria RAM se requieren 12 bits de address, por lo que se utiliza el bit A11 para seleccionar cada uno de los 2 bloques de 2 chip conectados en paralelo. Notar que por la posición en la que está implementada la RAM, en el comienzo los bits A13 y A12 valen ‘1’ y ‘0’ respectivamente.
- En toda la memoria implementada los bits de address 15 y 14 (resaltados en azul) valen ‘0’.

Para que la implementación que NO genere posiciones imagen en el espacio no implementado es necesario que para habilitar cada chip de memoria se consideren todos los bits de la dirección, por lo que la implementación tendrá la siguiente forma:


A[0..15]
16
A[0..13]
14
A[0..10]
11
D[0..7]
8
D[0..7]
A[14..15]
2
A[11..13]
3
Decodif. 3x8
A11
A12
A13
SEL0
SEL1
SEL2
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
000
001
010
011
100
101
110
111
enable
Decodif. 2x4
A14
A15
SEL0
SEL1
Q0
Q1
Q2
Q3
00
01
10
11
enable
ADDRs
DATA
EPROM
2Kx8bits
#1
CS
EPROM
2Kx8bits
#2
EPROM
2Kx8bits
#3
EPROM
2Kx8bits
#4
D[0..3]
4
D[4..7]
RAM
2Kx4bits
#1
RAM
2Kx4bits
#2
RAM
2Kx4bits
#3
RAM
2Kx4bits
#4

Respuesta B:


A[0..15]
16
A[0..10]
11
ADDRs
DATA
D[0..7]
8
D[0..7]
EPROM
2Kx8bits
#1
CS
A[11..13]
3
Decodif. 3x8
Q0
000
Q1
001
Q2
010
Q3
011
Q4
100
Q5
101
Q6
110
Q7
111
A11
SEL0
A12
SEL1
A13
SEL2
enable
EPROM
2Kx8bits
#2
EPROM
2Kx8bits
#3
EPROM
2Kx8bits
#4
D[0..3]
4
D[4..7]
RAM
2Kx4bits
#1
RAM
2Kx4bits
#2
RAM
2Kx4bits
#3
RAM
2Kx4bits
#4

Esta implementación es igual a la anterior, pero se ha eliminado el decodificador de 2x4 (el que decodificaba A15 y A14), dejando el decodificador de 3x8 siempre habilitado (conectado eléctricamente a '0'). En este caso, los chip de memoria estarán habilitados según corresponda por la combinación de los bits de address A13 a A0, sin importar los valores que tomen A15 y A14.
Con 'N' bits de address, tenemos 2^N combinaciones de dirección. En este caso, A15 y A14 pueden tomar valores '00', '01', '10' y '11'. Como se observa en el mapa de memoria dado, sólo la combinación '00' corresponde al espacio de memoria implementada, mientras que los otros tres casos generarán posiciones imagen.
Analizando el mapa de memoria resultante (abajo) se observa que el contenido de la EPROM y la RAM se replica 3 veces.