Taller 4
Memorias ROM

Objetivos

- Sustituir el diseño de circuitos combinatorios por una ROM cargada adecuadamente.
- Practicar la carga de ROM mediante programa.
- Practicar el uso de máscaras.

Introducción

Al momento de diseñar un circuito combinatorio podemos optar por el diseño clásico, como los implementados en el taller anterior, pero existen otras alternativas. En este taller se plantea la posibilidad de utilizar una memoria ROM que sustituya a la lógica combinatoria requerida para realizar una conversión de números representados en punto flotante a enteros representados en complemento a 2. Para lograr este objetivo debemos escribir un programa que cargue la ROM de forma adecuada.

Responder las siguientes preguntas:

a) En una memoria de 8Kx64:
- ¿Cuántas líneas son necesarias para direccionarla?
- ¿Cuál es el tamaño de palabra de la memoria?
- ¿Qué capacidad en bytes tiene dicha memoria?

b) A partir de la memoria de 8Kx64. ¿Es posible generar una memoria de 4Kx128, agregando sólo compuertas básicas y dispositivos de tercer estado? ¿Por qué?

Trabajo a realizar

Se desea utilizar una ROM para implementar una función que determine la parte entera de un número representado en Punto Flotante de Precisión Media (1 bit signo, 5 bits exponente, 10 bits parte fraccional). La parte entera a dar como salida deberá estar representada en Complemento a 2 de 16 bits. La función también deberá indicar, mediante un bit, si la operación no puede realizarse por no ser representable la parte entera en ese formato (salida de Overflow).

a) Determinar tamaño y organización de la ROM necesaria para implementar la función.
   1. Determinar el tamaño y la organización de la ROM
   2. Construirla en base a ROMs de 32K x 8 y 16K x 4 y compuertas básicas. Utilizar la menor cantidad de ROMs posible.

b) Escribir el programa que genera el contenido de la ROM. Para esta parte el lenguaje dispone solamente de aritmética entera y operaciones bit a bit (no dispone de aritmética de Punto Flotante).

Notas:

Se dispone de las siguientes operaciones para el tipo hexadecimal, con el fin de generar el contenido de la ROM.
- a | b (or bit a bit)
- a & b (and bit a bit)
- a >> b (shift *a* la derecha *b* lugares)
- a << b (shift *a* la izquierda *b* lugares)
- +, -, *, / y % (en complemento a 2)