EA869 – 1/2008 : Exemplo de decodificação

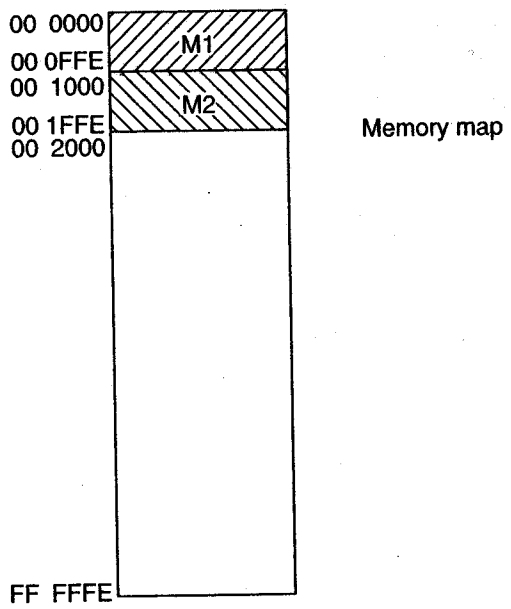
(fonte: Microprocessor Systems Design, Alan Clements, 3. ed, pgs. 310 – 312)

Now let's look at a more practical example of full address decoding. A 68000 microcomputer requires 10 K-words of ROM (arranged as a block of 2 K-words plus a block of 8 K-words), 2 K-words of random access read/write memory, two words for peripheral 1, and two words for peripheral 2. We will call the five memory devices to be decoded ROM1 (2K), RAM (2K), ROM2 (8K), PERI1 (2) and PERI2 (2). Each device has an active-low chip-select input and may be located anywhere within the system's memory map, with the sole exception of ROM1, which must respond to addresses in the range $00 0000 to $00 0FFF.

The first step in solving this problem is to construct an *address table*. Such a table is given in Table 5.1, where the vertical columns represent the 23 address lines A_{01} to A_{23}, and the rows represent the five memory components to be decoded. An X in an address column means that the address line takes part in the selection of a location within the specified component. A 1 or 0 in an address column means that the address line must be 1 or 0 to select that component.

The address lines to be decoded for each memory component must be selected as either zero or one. How this is done is, to a certain extent, unimportant. In Table 5.1, we selected the lowest possible address for each device. All that needs to be done to achieve full address decoding is to decode every address line that does not select a location within a device and to ensure that no two devices can be accessed simultaneously.

Figure 5.5 describes a suitable address decoder for the system of Table 5.1. The two peripherals each require 22 address lines to be decoded, as only one address line, A_{01},



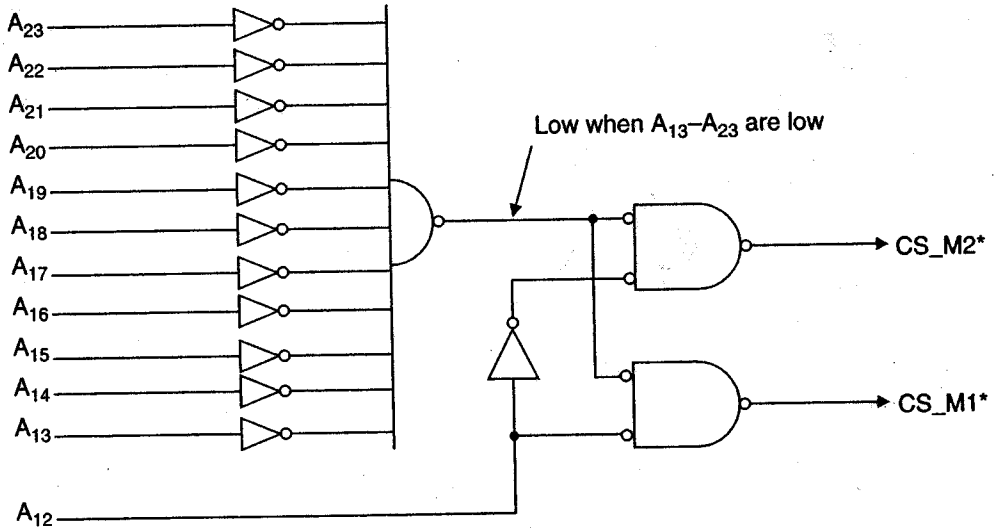


Figure 5.4 Resolving the conflict between the memory components by full address decoding

selects a location (one of two) within a peripheral. PERI1 is selected whenever A_{16}–A_{23}, A_{02}–A_{14} are all zero, and A_{15} is a logical 1. Two 74LS133 13-input NAND gates perform the bulk of the address decoding. Whenever the outputs of both NAND gates are simultaneously low, the CS* output of the 3-input AND gate goes low to select PERI1.

Table 5.1 Address decoding table illustrating full address decoding

Device	A_{23}	A_{22}	A_{21}	$A_{20}\ldots A_{15}$	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1
ROM1	0	0	0	0...0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
RAM	0	0	0	0...0	0	0	1	X	X	X	X	X	X	X	X	X	X	X
ROM2	0	0	0	0...0	1	X	X	X	X	X	X	X	X	X	X	X	X	X
PERI1	0	0	0	0..01	0	0	0	0	0	0	0	0	0	0	0	0	0	X
PERI2	0	0	0	0..01	0	0	0	0	0	0	0	0	0	0	0	0	1	X

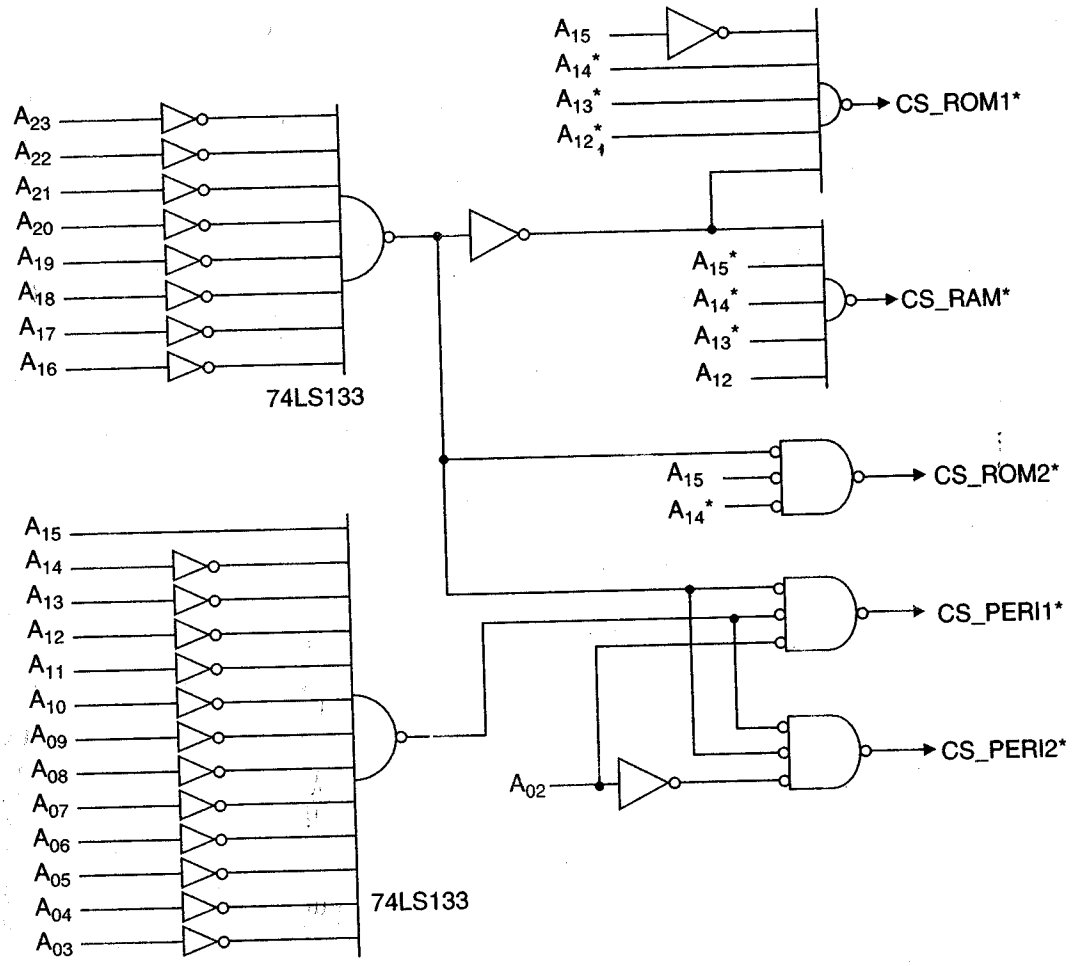


Figure 5.5 Full address decoding network corresponding to Table 5.1

The circuit of Figure 5.5 highlights a paradox of microcomputer design. The microcomputer and peripheral are available as two single chips, and yet the address decoding circuit of Figure 5.5 requires a total of eight chips (assuming that four hex-inverters are used).