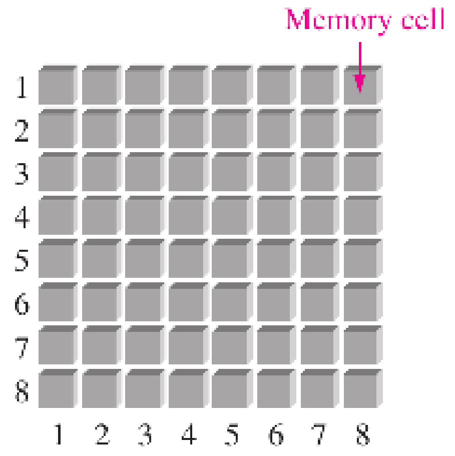
EC2.101 – Digital Systems and Microcontrollers

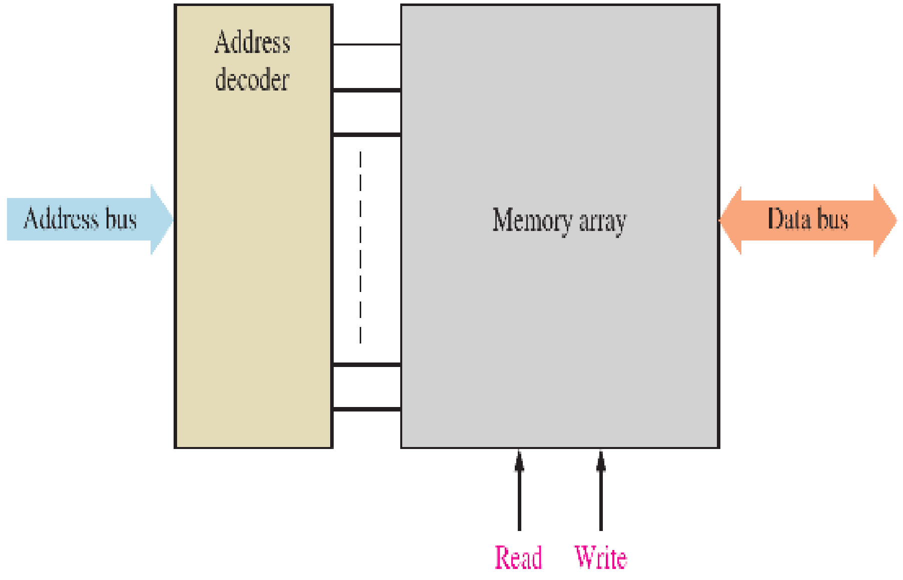
Lecture 25 – Memory architecture 2

Chapter 7

Address decoding: coincident decoding

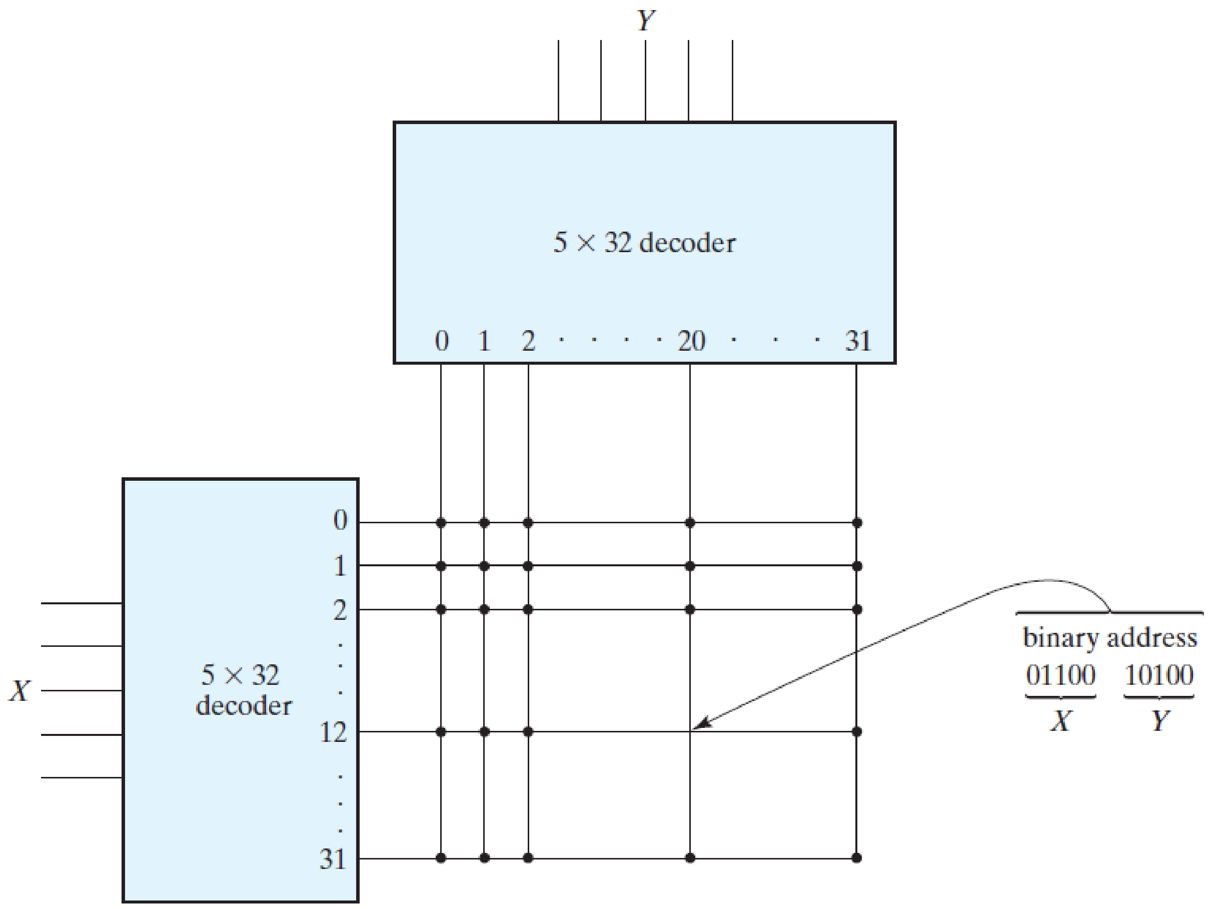
- A decoder with *k* inputs and 2^k outputs requires 2^k AND gates with *k* inputs per gate
- The total number of gates and the number of inputs per gate can be reduced by employing *two decoders in a two-dimensional selection scheme*
  - The basic idea in two-dimensional decoding is to arrange the memory cells in an array that is close as possible to square
  - In this configuration, two *k* /2-input decoders are used instead of one *k* -input decoder
  - One decoder performs the row selection and the othe the column selection in a two-dimensional matrix configuration





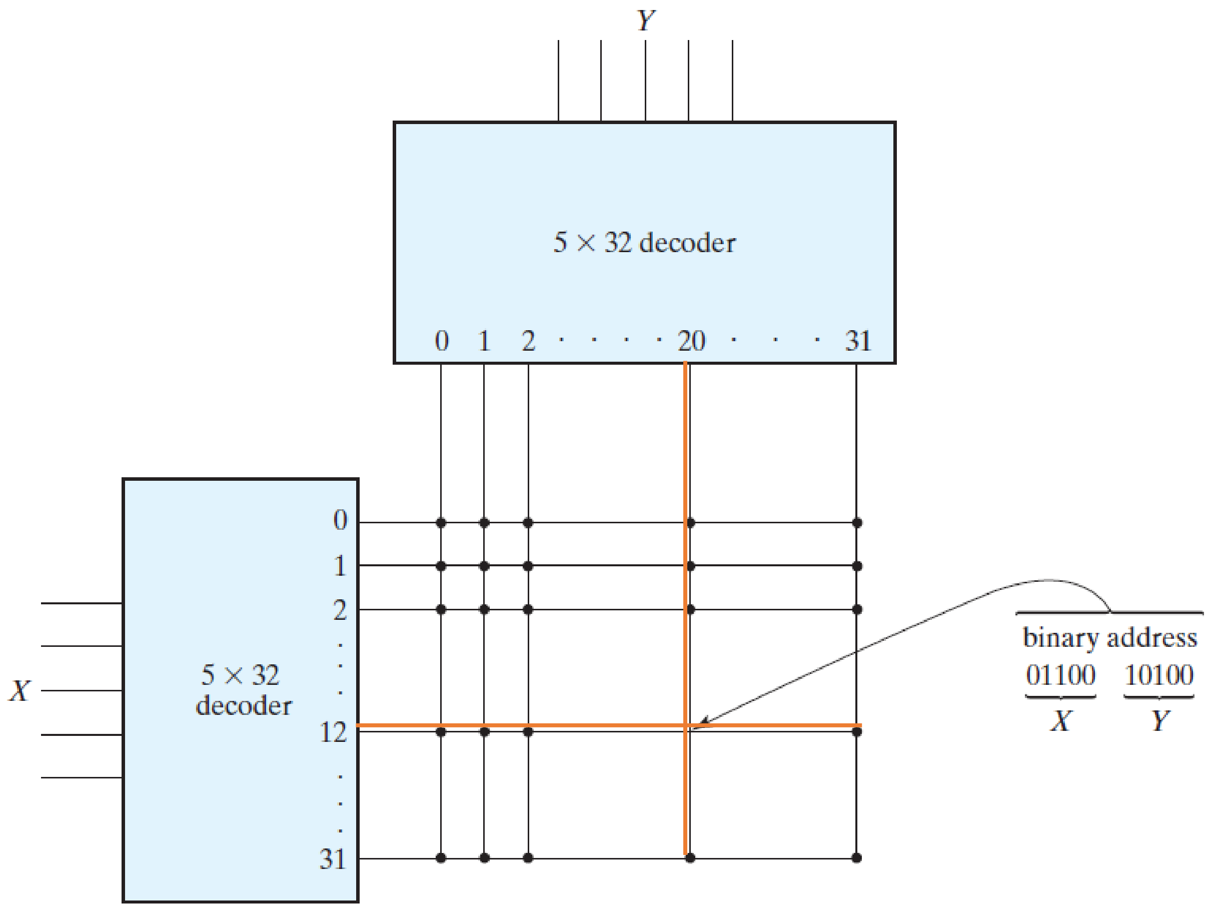
Coincident decoding

- For example, instead of using a single 10 * 1,024 decoder, we use two 5 * 32 decoders
  - With the single decoder, we would need 1,024 AND gates with 10 inputs in each
  - In the two-decoder case, we need 64 AND gates with 5 inputs in each
- The five most significant bits of the address go to input *X* and the five least significant bits go to input *Y*
- *Each word within the memory array is selected by the coincidence of one X line and one Y line*
- Thus, each word in memory is selected by the coincidence between 1 of 32 rows and 1 of 32 columns, for a total of 1,024 words



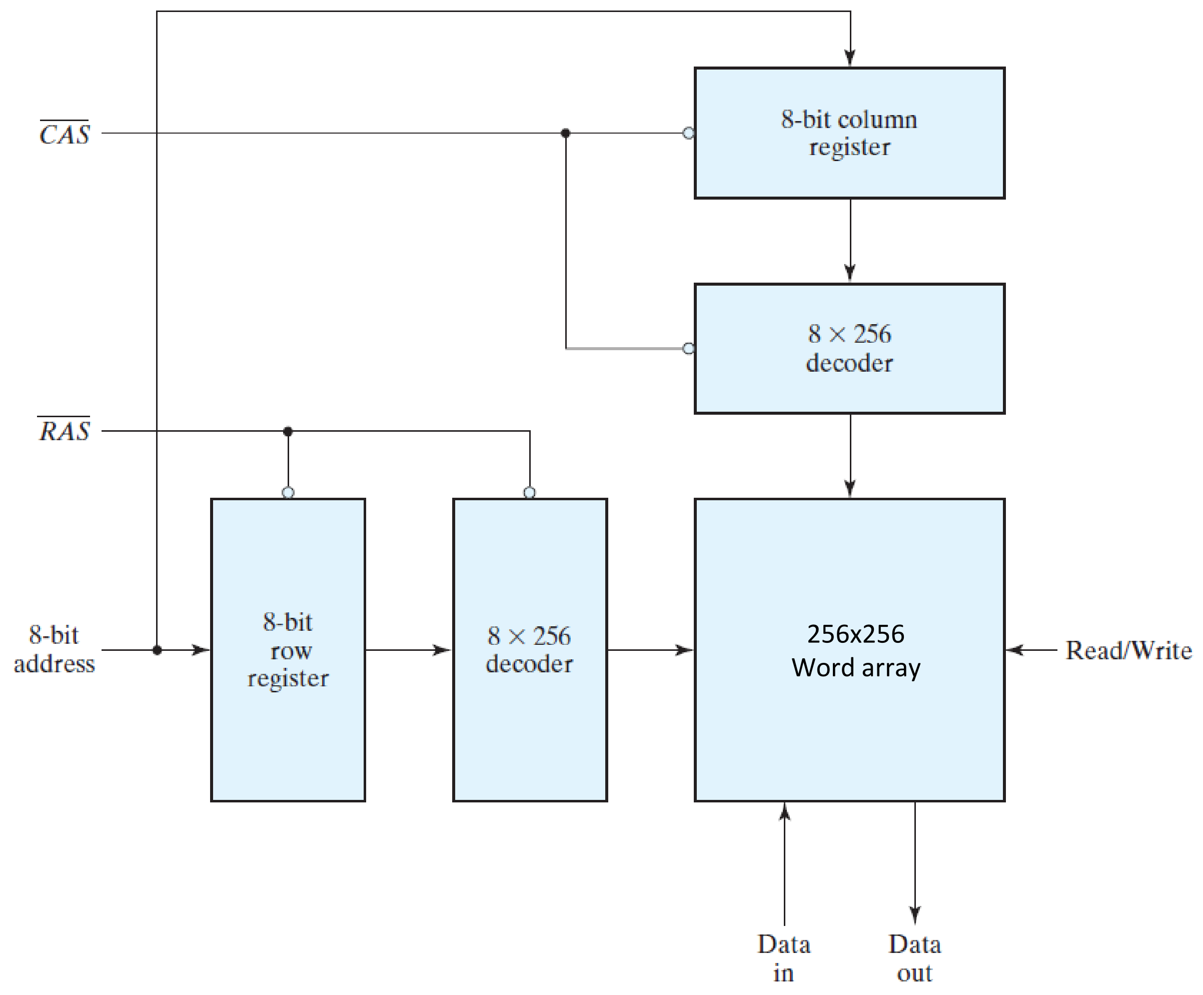
Coincident decoding

- As an example, consider the word whose address is 404. The 10-bit binary equivalent of 404 is 01100 10100. This makes *X* = 01100 (binary 12) and *Y* = 10100 (binary 20)
- The *n* -bit word that is selected lies in the *X* decoder output number 12 and the *Y* decoder output number 20
- All the bits of the word are selected for reading or writing



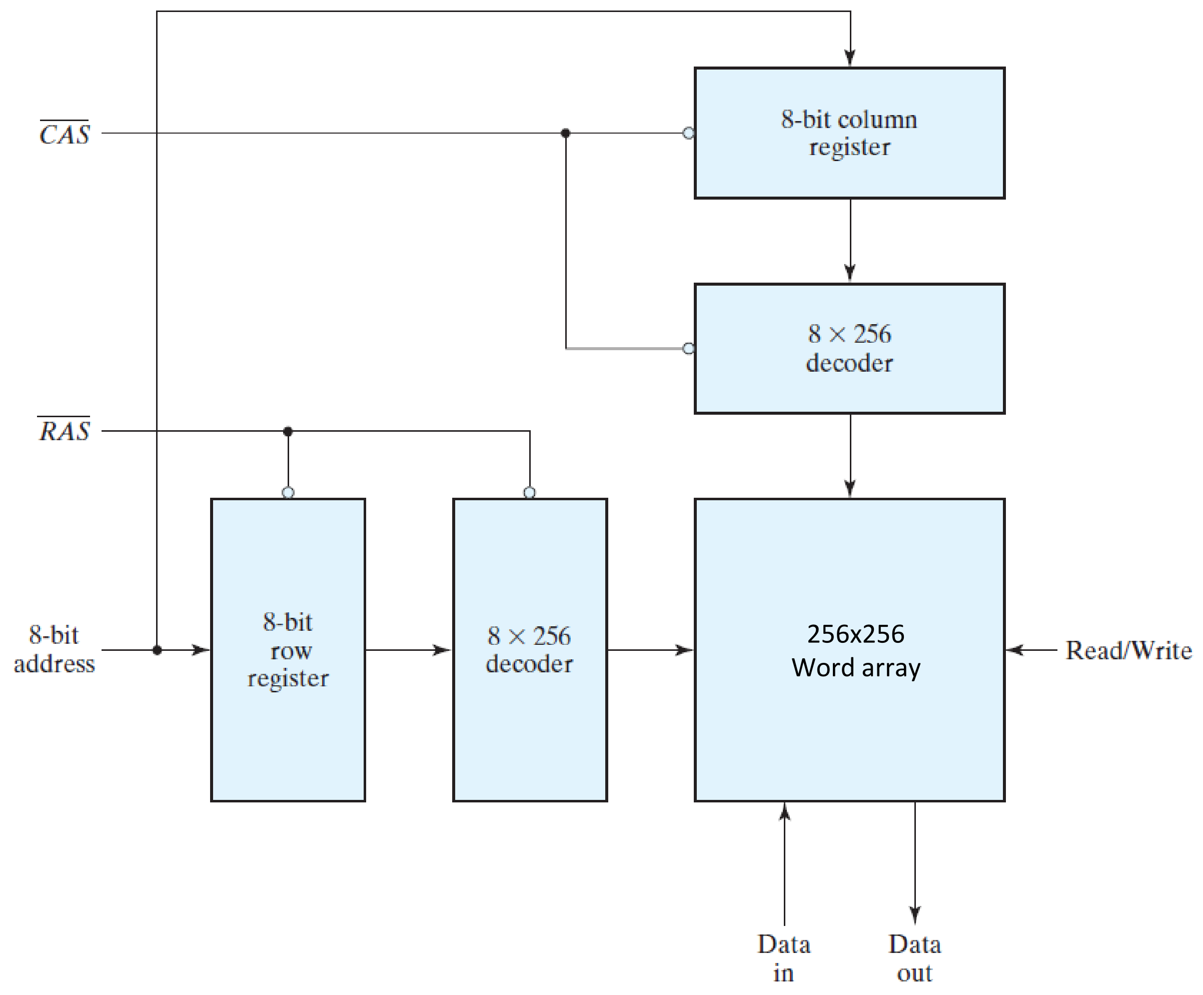
Address multiplexing

- To reduce the number of pins in the IC package, designers utilize address multiplexing whereby one set of address input pins accommodates the address components
- In a two-dimensional array, the address is applied in two parts at different times, with the row address first and the column address second
- Since the same set of pins is used for both parts of the address, the size of the package is decreased significantly



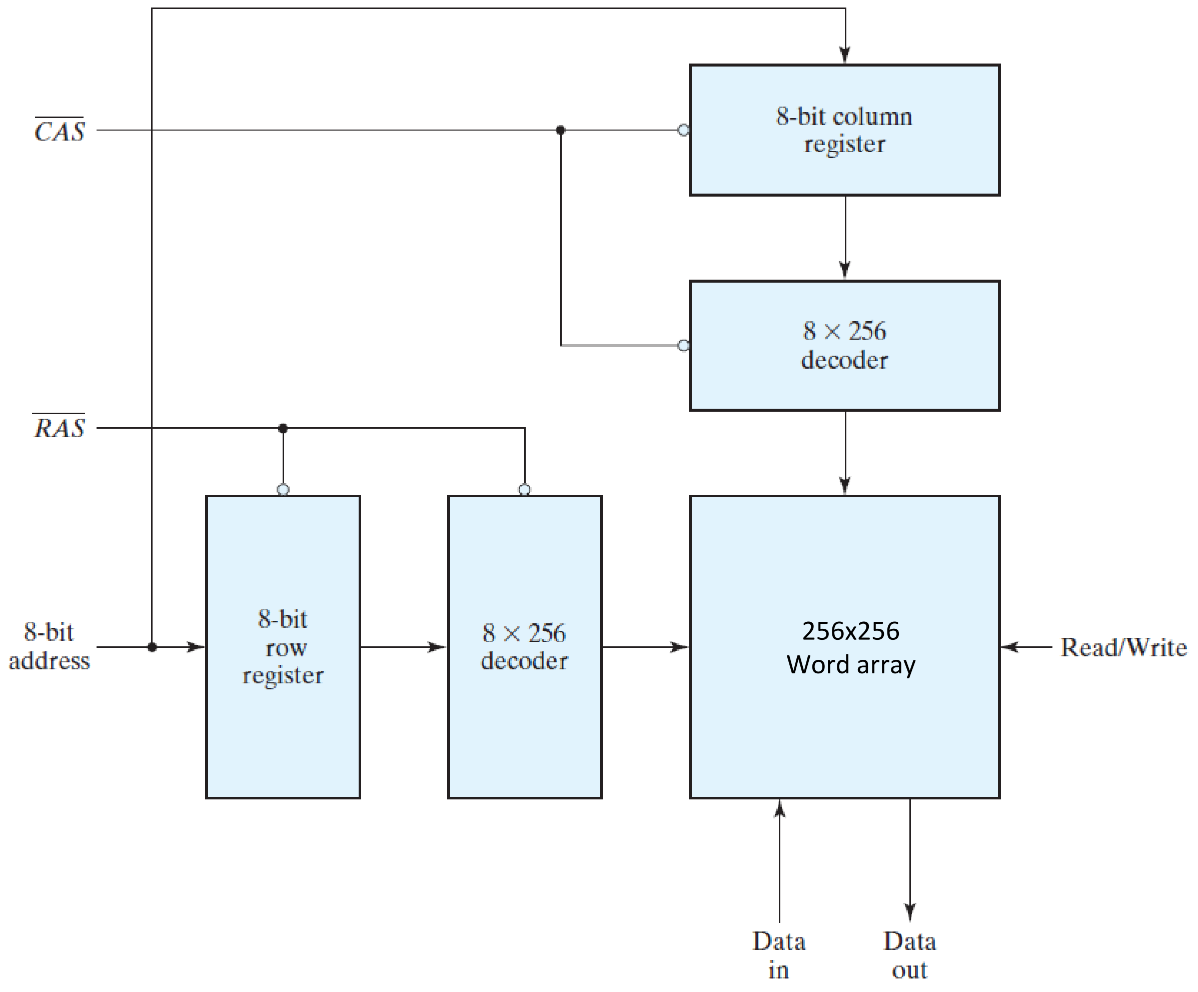
Address multiplexing

- We will use a 64K-word memory to illustrate the address-multiplexing idea
- The memory consists of a two-dimensional array of cells arranged into 256 rows by 256 columns, for a total of $2^8 * 2^8 = 2^{16}$ = 64K words
- There is a single data input line, a single data output line, and a read/write control, as well as an eight-bit address input and two address *strobes,* the latter included for enabling the row and column address into their respective registers
- The row address strobe (RAS) enables the eight-bit row register, and the column address strobe (CAS) enables the eight-bit column register



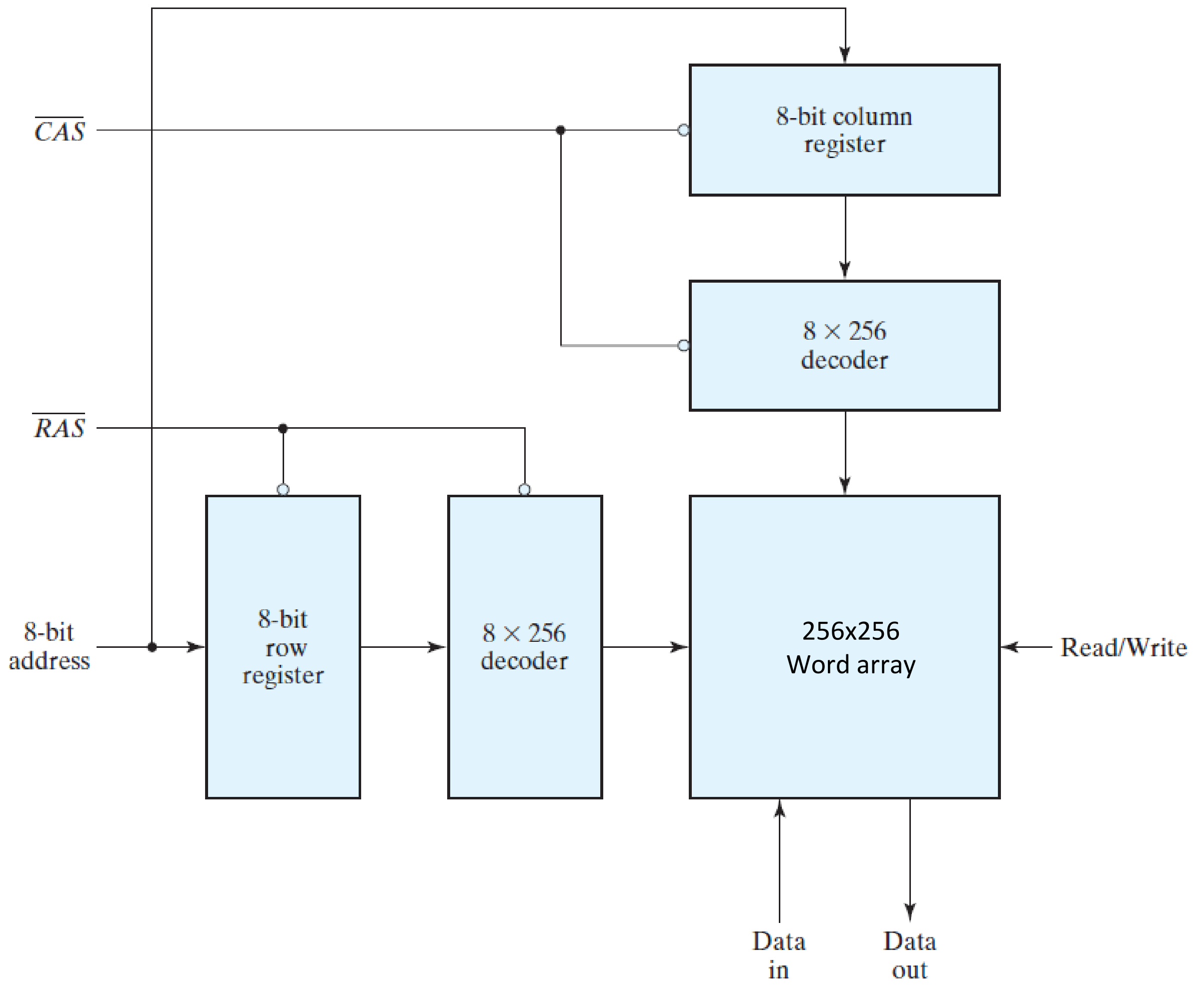
Address multiplexing

- The 8-bit row address is applied to the address inputs and RAS is activated
- This loads the row address into the row address register
- RAS also enables the row decoder so that it can decode the row address and select one row of the array



Address multiplexing

- The 8-bit column address is again applied to the address inputs, and CAS activated
- This transfers the column address into the column register and enables the column decoder
- Now the two parts of the address are in their respective registers, the decoders have decoded them to select the one cell corresponding to the row and column address, and a read or write operation can be performed on that cell
- CAS must go back to the logic 1 level before initiating another memory operation



Error detection and correction

- The dynamic physical interaction of the electrical signals affecting the data path of a memory unit may cause occasional errors in storing and retrieving the binary information

- The reliability of a memory unit may be improved by employing error-detecting and error-correcting codes

- The most common error detection scheme is the *parity bit*

The parity bit

- A parity bit is generated and stored along with the data word in memory
- The parity of the word is checked after reading it from memory
- The data word is accepted if the parity of the bits read out is correct
- If the parity checked results in an inversion, an error is detected
- However, it cannot be corrected
- Error correction requires more complex mechanisms such as the *Hamming code*

The Hamming code

- One of the most common error-correcting codes used in RAMs was devised by R. W. Hamming
- In the Hamming code, *k* parity bits are added to an *n* -bit data word, forming a new word of *n* + *k* bits
  - The bit positions are numbered in sequence from 1 to *n* + *k*
  - Those positions numbered as a power of 2 are reserved for the parity bits
  - The code can be used with words of any length
- Consider, for example, the 8-bit data word 11000100
- We include 4 parity bits with the 8-bit word and make a 12 bit word

While storing:

Bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

$$P_1 = \text{XOR of bits } (3, 5, 7, 9, 11) = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$$
$$P_2 = \text{XOR of bits } (3, 6, 7, 10, 11) = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$$
$$P_4 = \text{XOR of bits } (5, 6, 7, 12) = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$
$$P_8 = \text{XOR of bits } (9, 10, 11, 12) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

While reading:

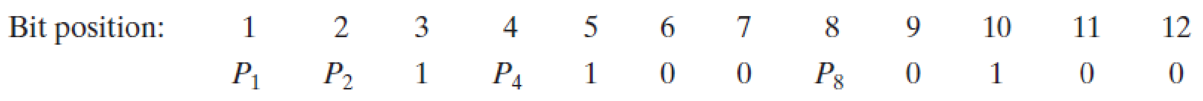
	0	0	1	1	1	0	0	1	0	1	0	0
Bit position:	1	2	3	4	5	6	7	8	9	10	11	12

$$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$$
$$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$$
$$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$$
$$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$$

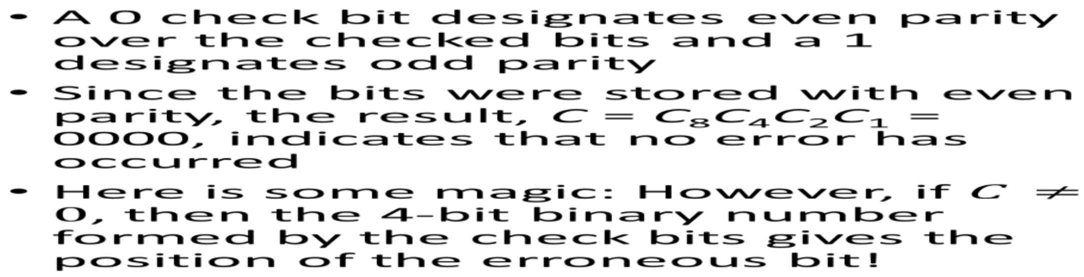
The Hamming code

- A 0 check bit designates even parity over the checked bits and a 1 designates odd parity
- Since the bits were stored with even parity, the result, $C = C_8C_4C_2C_1$ = 0000, indicates that no error has occurred
- Here is some magic: However, if $C \neq 0$, then the 4-bit binary number formed by the check bits gives the position of the erroneous bit!

While storing:

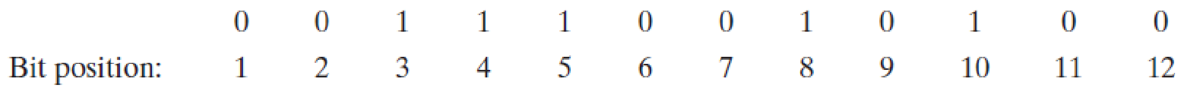


Bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0



- A 0 check bit designates even parity over the checked bits and a 1 designates odd parity
- Since the bits were stored with even parity, the result, $C = C_8C_4C_2C_1$ = 0000, indicates that no error has occurred
- Here is some magic: However, if $C \neq$ 0, then the 4-bit binary number formed by the check bits gives the position of the erroneous bit!

While reading:



	0	0	1	1	1	0	0	1	0	1	0	0
Bit position:	1	2	3	4	5	6	7	8	9	10	11	12

$$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$$

$$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$$

$$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$$

$$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$$

The Hamming code

Detecting the position of erroneous bit

Bit position:	1	2	3	4	5	6	7	8	9	10	11	12	
	0	0	1	1	1	0	0	1	0	1	0	0	No error
	$\underline{1}$	0	1	1	1	0	0	1	0	1	0	0	Error in bit 1
	0	0	1	1	$\underline{0}$	0	0	1	0	1	0	0	Error in bit 5

	C_8	C_4	C_2	C_1
For no error:	0	0	0	0
With error in bit 1:	0	0	0	1
With error in bit 5:	0	1	0	1

$$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$$

$$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$$

$$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$$

$$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$$