Reliability of digital systems

Continuous diagnostics - control circuits of the memory

Lecture notes for course PA175/13-1

Lesson 13 - content

13 Checking the correct operation of the memory of Digital Systems

Digital system memories are among the modules with a high probability of failure. This probability is not given by the intensity of memory component failures, but also by the mode of operation. Mainly for this reason, even for electronic devices with the lowest demands on the correctness and reliability of operation, means for checking the correctness of operation are installed.

While other parts of the digital system create intermittent faults that can be eliminated in a relatively simple manner, memories have the property that a temporarily erroneous logic signal level is fixed in them. With this mechanism, the occasional failure becomes a permanent failure, and due to the requirements for reliability of operation and common digital systems, it is desirable to prevent these problems.

ROM and RAM use a variety of technologies in digital systems. One of the most commonly used technologies is dynamic RAM - referred to as DRAM. We will describe the properties of this type of memory in more detail.

13.1 Faults of memory cells

DRAM failures can be divided into two groups:

- permanent failures,
- occasional failures.

Permanent failures are caused by mechanical damage to the structure of the memory chip, i.e. a defect in the memory matrix itself (bit cells of the memory), in the connections in the memory matrix, in the read and write amplifiers or in the supporting logic. The occurrence of a failure can be varied, from errors already in production through thermal or electrical breakdown to physical damage to the memory chip. Specifically, the failure is of the type:

- defect to the memory cell,
- defect to connections in the memory cell matrix,
- defect of read and write amplifiers,
- memory chip control circuit defects.

Occasional failures arise due to external conditions and after they disappear, the affected part of the memory works flawlessly again. Thus, occasional failures will damage the information stored in the memory, but will not cause mechanical - irreversible - damage to the structure of the memory chip. The typical causes of occasional memory failures are:

- influence of radioactive radiation - impact of alpha particles into the structure of the memory chip,
- exceeding the recommended operating temperature range,
- deviations of input signal levels,
- non-compliance with signal timing,
- interference, crosstalk and reflections on impedance mismatch lines (corrosion, mechanical impurities).

13.3.1 Dynamic memory cell failures

To illustrate the interpretation of the failures of dynamic memory cells, it is appropriate to approach their implementation. The DRAM memory cell is implemented by one planar transistor and a capacitor. The capacitor can again be made by planar technology or by a capacitor formed in the notch of the chip substrate. The planar capacitor occupies a large area on the chip compared to the active elements, and therefore the structure of the capacitor is implemented in the depth of the chip. Its capacity is several pF. The capacitor created in this way is electrically separated from the surroundings by an insulating layer - mostly silica (SiO_2). The insulating layer is characterized by a high but measurable impedance – see Figure 13.1. Due to this leakage, the memory capacitor is discharged, and therefore it is necessary to restore the charge in the memory capacitor. Refreshing is done by reading and rewriting the information.

Memory cells show both permanent and occasional failures - but both have a fatal effect on the function of the digital system as a whole.

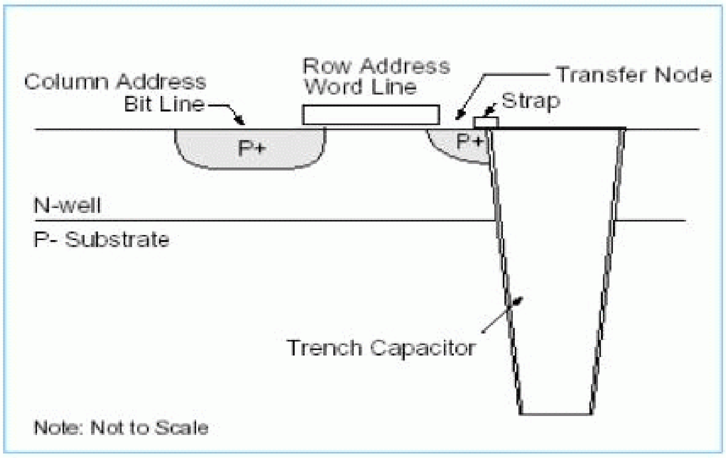

Figure 13.1: Scheme cut DRAM memory cell

Permanent memory cell failures are caused by damage to its components or interactions between memory cells. Failure of the memory cell can be caused by a short circuit in the memory cell, breakdown of the capacitor isolation layer, disconnection of the memory cell from the control transistor due to too high voltage or thermal destruction of the chip, etc. Their manifestation can be described by a disorder model of type $\mathbf{t_0}$ or $\mathbf{t_1}$.

The interaction of neighboring memory cells is most often caused by capacitive coupling. These connections manifest themselves in such a way that, for example, if an inverse value is written to four or more topologically adjacent cells around the test cell, the original value is overwritten in the test cell. This failure cannot be detected by simple memory tests, it is necessary to apply tests aimed at testing parasitic interactions - such as the Galpat test.

Occasional memory cell failures are most often caused by radiation. Even in otherwise sterile conditions, the environment shows a certain intensity of radioactive radiation - the radioactive background. Neutron radiation and radiation produced by isotopes contained in materials and raw materials used in the production of not only digital systems contribute to it. The memory chip itself and its packaging therefore contain various isotopes, even with very careful production and selection of raw materials, which, with other forms of radiation, also emit alpha particles - relatively slow-moving helium nuclei - with low intensity. When the alpha particle hits the structure of the memory cell capacitor, its charge is discharged faster. As the component density on the memory chip increases, the probability of a memory capacitor hitting decreases (the memory cell has a smaller surface area and thus represents a smaller target). On the other hand, increasing the integration (number of memory cells per mm2), decreasing the supply voltage and decreasing the capacitor capacity increases the probability of premature discharge of the memory capacitor by interactions with alpha particles.

Cosmic radiation in the upper atmosphere causes similar effects. It is known from experience that at altitudes of around 10 km, an occasional memory failure occurs every 5 hours in the memory of a standard 256 MB laptop.

13.3.2 Faults in the address decoder and read amplifiers

This is another relatively common memory failure. The memory cell can be addressed linearly or coincidentally. The address is interpreted in binary code in digital systems, with the address decoders converting the binary code to **1zN** code. If an address decoder failure occurs, either several memory cells are read or written simultaneously or no memory cell is selected.

An address decoder failure can be a permanent or occasional. The permanent failure is caused by damage to the address decoder components. The occasional failure of the address decoder is caused by unwanted signal interactions due to capacitive couplings between the individual selection wires. The frequency of intermittent address decoder faults may depend on the supply voltage or the sequence of consecutive addresses (this fault is very difficult to locate).

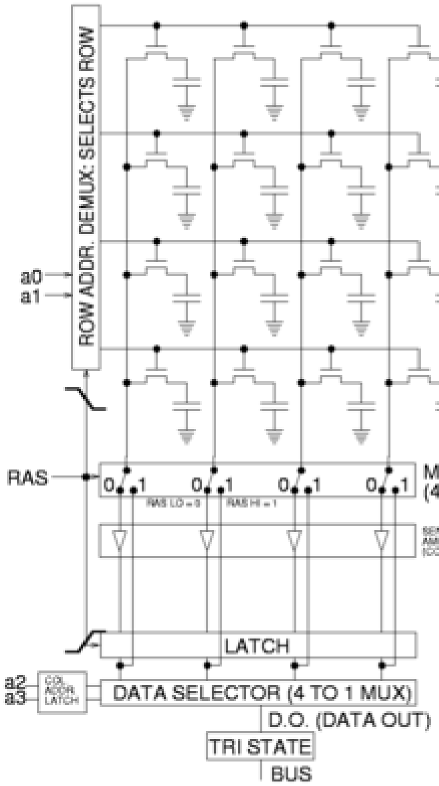

Figure 13.2: The block of memory cells in the DRAM

13.3.3 Failures caused by incorrect timing

Incorrect timing allows many phenomena to affect stored or read information. These phenomena most often do not cause the destruction of the components of the memory chip - so they are mostly occasional failures. Undesirable phenomena include the interaction of differential amplifiers. Another phenomenon is a phenomenon called data amplifier memory - saturation of some stage of the data amplifier. Amplifiers require a longer recovery time after reading one logical value repeatedly; reading the first opposite logical value may not be successful. Another problem is the large capacities between the memory matrix connections - their suppression requires an increase in access time, especially when the temperature of the memory chips increases.

Note:

To increase the access time, it is necessary to perform tests of memory chips on a loaded and long-running computer, when the temperatures of the memory chips stabilize.

As the temperature increases, the parasitic time constant of the memory capacitor decreases, which forces an increase in the refresh rate.

13.3.4 Faults caused by unsuitable supply voltage parameters

Suitable supply voltage parameters are - supply voltage amplitude and supply voltage ripple.

Most memory chips require strict adherence to the nominal amplitude of the supply voltage. Some types of chips work reliably even with small deviations of the supply voltage from the nominal, but this is not the rule. The sensitivity of dynamic memories to the amplitude of the supply voltage is given by the principle of operation of memory cells and the sensitivity of the parameters of data amplifiers to this parameter.

Perhaps even more sensitive are the memory chips to various oscillations and ripple of the supply voltage that can occur in different places in the computer - starting by the CD-ROM engine and ending the microprocessor to sleep mode. The power supply also does not have to filter out all external influences, especially those that have a high frequency (synchronous electric motors, switching on the fluorescent lamp, etc.). The nature of the errors thus generated is typically random (they do not affect only one group of memory cells) and in some cases also irregular.

13.4 Memory control circuits

Occasional failures are the main reason for the implementation of control circuits in the memory of digital systems. In more demanding implementations, both memory content failure indication and circuitry for fixing some types of failures are applied. The most commonly used types of memory activity control are:

- parity,
- Hamming codes,
- checksums,
- correction routines.

13.4.1 Parity

This is the simplest method of checking memory activity. This principle uses the redundancy of information often within a single byte. Redundancy bit informs the system whether a selected number of bits preserve the original number. If the controlled words invert two bits, the parity does not indicate adverse condition. It is thus obvious that parity data protection applications are very limited.

Parity is used both for creating security bits of information transmitted and stored in digital systems, as well as for checking secure information. Parity is an additional bit, which complements the information bits even or odd number of ones. Most often generates parity for byte and odd parity is used when a binary zero in the parity bit is 1 and it is clear that the data source is working (if it had been disconnected, they will have all the bits including the parity value **0**), see next picture.

b_0	**b_1**	**b_2**	**b_3**	**b_4**	**b_5**	**b_6**	**b_7**	**P**

a)

0	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**

b)

0	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**

c)

Figure 13.3: Principle and example of parity security implementation
a) principle of redundancy security,
b) example of odd parity security,
c) example of event parity security.

The above diagram represents a linear parity word (here byte), the change is detected by only one bit. This means that a simple inversion of two bits of parity already detect.

13.4.2 Checking memory by Hamming codes

Hamming code creates parity based on different bits of the word. The failed bit should generate code expressing its position in the word. Hamming code is essentially a distributed parity over a protected word. Depending on the number of check bits added, this code is able to increase the code distance according to the needs of the application. In practice, a code with a code distance of 3 is used for words 64 or 128 bits long.

Individual bits of the security code $\mathbf{p_i}$ are generated as even parity bits by **XOR** circuits - represented here by the symbol ⊕ - of the corresponding data bits $\mathbf{a_i}$.

$$\mathbf{p_1 \oplus a_1 \oplus a_2 \oplus a_4 = 0}$$

To increase the effectiveness of Hamming code was created so-called *extension of the binary Hamming code*, which is based on adding additional checking bit $\mathbf{p_T}$ to the beginning of each coded word. This additional bit is used for control of the word parity. This bit is chosen so that all bits of the secured word form as the even parity. The extended code allows to correct one failure and also detects two failures within the transmitted word.

$$\mathbf{p_T \oplus b_i \oplus b_j \oplus b_k \oplus \dots\dots\dots\dots . \oplus b_n = 0}$$

Each time data is read from memory, syndrome bits are generated, the value of which captures the correctness of the content of the read word. Syndrome bits are generated according to the following equations:

$$\mathbf{s_i = \sum_{i=1}^{n} (p_i + b_{C_i})_{mod\,2}}$$

$$\mathbf{s_T = (\sum_{j=0}^{n} (p_j + b_j)_{mod\,2} + p_T)_{mod\,2}}$$

The share of individual data bits to create Hamming code is shown below.

We propose a method of locating an faulty bit in the word. Hamming code creates parity based on different bits of the word. Failed bit should generate code expressing the position of the faulty bit in the word.

The unsecured **m** bits **a_i** of the word **A** with a code distance **1**

It is joined on **k** other bits **p_1**, **p_2**, ……. **p_k** of the control word **P**, which consists from unique, non-repeat subsets bits **a_i**. Secured word **B** formed by bits **b_1**, **b_2**, ..…...... **b_{m+k}** we get by the appropriate grouping of bits **a_i** and **p_i**.

Above is secured word **B** is possible form a number **S** consisting from **syndrome bits s_1**, **s_2**, ….**s_k** (*syndrome of failure*), whose numerical value directly indicate the order of bit errors in the secured word **B**.

Note:

Syndrome gives the position of all the wrong bits secured word and of the parity bits of course.

Syndrome reflects the position of bit error with a faulty value, and therefore value **0** corresponds to flawless code and the values from **1** to **m + k** corresponds to locate ok the failed bit in the secured word. Because it is necessary to cover a total of **m + k +1** option, the number of bits of security to meet the inequality:

$$\mathbf{2^k \geq m+k+1}$$

The procedure of generating control bits can be described as follows:

- position of equal power of the resulting code used for the unique parity bits **p_i**, where **i** takes values (**1, 2, 4, 8, 16, 32, ...**)
- other positions are allocated into unsecured bits of the words **a_i**, where **i** takes values (**3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, ...**)
- each parity bit is calculated from a unique combination of bits of the insecure word as the even parity. Location of the parity bit clearly specifies which sequence of bits is used for generating the parity bit.

Parity bit **p_1** (first position) belongs to all the **odd bits** from the information bit **a_1** (**3rd**) of the secure word and **a_2** (**5th)** bit etc. of the secured word.

Parity bit **p_2** (second position) is the security bit of each **odd pair bits** from the secured word starting with **a_3** (**6th**) and **a_4** (**7th**) bits etc. of the secured word.

Parity bit **p_3** (fourth position) provides each **odd four bits of** the secured word starting with bits **5th**, **6th**, **7th** and **8th** etc. of the secured words. For additional parity bits are proceeding by analogy.

Protection of **64-bit words** using Hamming code is given below. For secure is used here **8 bits** – it presents an extended protection. Minimal numbers of Hemming code bits are **7 bits**.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
C_0 (1)	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																															
C_1 (2)		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨	▨	▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨
C_2 (4)			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨
C_4 (8)					▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨
C_8 (16)									▨	▨	▨	▨	▨	▨	▨	▨									▨	▨	▨	▨	▨	▨	▨	▨									▨	▨	▨	▨	▨	▨	▨	▨									▨	▨	▨	▨	▨	▨	▨	▨
C_{16} (32)																	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
C_{32} (64)	▨																																▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
C_T	🟩	🟩	🟩		🟩			🟩	🟩			🟩		🟩	🟩		🟩			🟩		🟩	🟩			🟩	🟩		🟩			🟩		🟩	🟩		🟩			🟩	🟩			🟩		🟩	🟩		🟩			🟩		🟩	🟩			🟩	🟩		🟩			🟩
	65	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	51	53	55	57	59	61	63	67	66	68	70	72	74	76	78	80	82	84	86	88	90	92	94	96	98	100	102	104	106	108	110	112	114	116	118	120	122	124	126

Table 13.1: Generating matrix

file

syndrom bits $S_0 \approx S_{32}$	*S_T*	*status read word*
00000000	**0**	**error free**
≠ 0	**1**	**single error**
≠ 0	**0**	**double faulted**
00000000	**1**	**error checking bits**

Table 13.2: The meaning of syndrome bits

S_1				1	0	1	0	1	0	1	0	1
S_2				2	0	0	1	1	0	0	1	1
S_4				4	0	0	0	0	1	1	1	1
S_{32}	S_{16}	S_8	S_0	bit byte	0	1	2	3	4	5	6	7
8	4	2	1									
0	0	0	0		p_T	p_1	p_2		p_4			
0	0	0	1	1	p_0	1	2	3	4	5	6	7
0	0	1	0		p_8							
0	0	1	1	3	8	9	10	11	12	13	14	15
0	1	0	0		p_{16}							
0	1	0	1	5	16	17	18	19	20	21	22	23
0	1	1	0									
0	1	1	1	7	24	25	25	27	28	29	30	31
1	0	0	0	8	p_{32}	33	34	35	36	37	38	39
1	0	0	1		0	32						
1	0	1	0	10	40	41	42	43	44	45	46	47
1	0	1	1									
1	1	0	0	12	48	49	50	51	52	53	54	55
1	1	0	1									
1	1	1	0	14	56	57	58	59	60	61	62	63
1	1	1	1									

Table 13.3: Table localization of the single error

13.5 Control Numbers

Checksum is different from the arithmetic sum transfers to neglect higher order, is actually the lowest order of the arithmetic sum, or add it to the selected value. If we do not only error detect, but also require the error correction must be the check digit combined with other security mechanisms to enable error correction to make, see the following example.

Example:

1	2	4	5	8	6	0	0	0	0	0	0	6
1	2	4	5	8	6	0	0	0	0	0	0	4

1+2+4+5+8+6+ …….. +0 = 26 → direct in 10 = 6

1+2+4+5+8+6+ …….. +0 = 26 → complement into 10 = 4

1	2	4	5	X	6	0	0	0	0	0	0	6
1	2	4	5	X	6	0	0	0	0	0	0	4

1+2+4+5+6+0 …….. +0=18 → complement in 10 = **2**, damaged digits = **(2+6)= 8** **low significant position**
1+2+4+5+6+0 …….. +0+4=22 → the value of the damaged number is complemet in 10 = **8**

13.6 Repeated writing and reading method

Repeated writing and reading method use these property failures of memory chips. Procedure for the suppression of influence failures is as follows:

- involvement of registry data of memory such as counters and reset its contents,
- first reading the word from memory cell and *signalization of the double error*,
- first write *inverted* word with failed bits to the same memory cell,
- second reading – content is a inverted word from the same memory cell with failed bits,
- registration faulty bits of memory cells into register-counter. Properly right bits of the word contain the **log.1**. Failure bits contain the **log.0**,
- second write – (inverted) firstly write word into the same memory cell,
- third reading the word from the same memory cell,
- inverting bits of the read words by registering faulty bits.

	true data bit	error log.1	true data bit	failed log.0
original data	0	0	1	1
fist read from memory – signalization of double fault	0	1	1	0
first write - *inverted* word	1	0	0	1
second read from the same memory address	1	1	0	0
content of the data register-counter (0 indicates the failed bit)	1	0	1	0
second write – *inverted* word from first read	1	0	0	1
third read from the same memory address	1	1	0	0
content of the data register after third read from memory	0	1	1	0
inversion of the failure bits	0	0	1	1

Table 13.4: Algorithm of the repeated writing and reading method

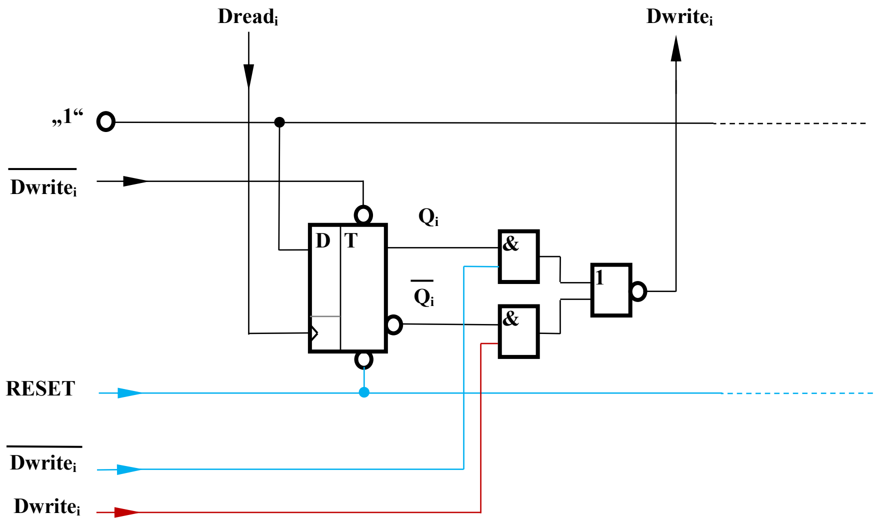

Figure 13.4: A schema of one bit of the memory data memory adapted to repeated writing and reading method

Implementing this method requires modifying memory controller algorithms and modifying the structure of the memory data register. The registry of the memory data during the repair routine in the **READ** operation acts as a counter. If the data register is also used for **WRITE**, asynchronous input is used for data writing. Before each repair operation, the data log must be reset asynchronously. The schema of one bit of the data memory registry see Figure 13.4.

Repeated writing and reading method is often complements the Hamming code check and uses the correct memory content from the locations where the portion of the software that cannot be replaced by a copy at that time is stored, and which, after recovery, will then be deployed to a different, flawless area of memory space.

Repeated writing and reading method is applicable in the absolute addressing phase. The content of the broken bit is inverted depending on the contents of the double-bit register during transmission on the system data bus. Block Diagram of double error repair see – Figure 13.5

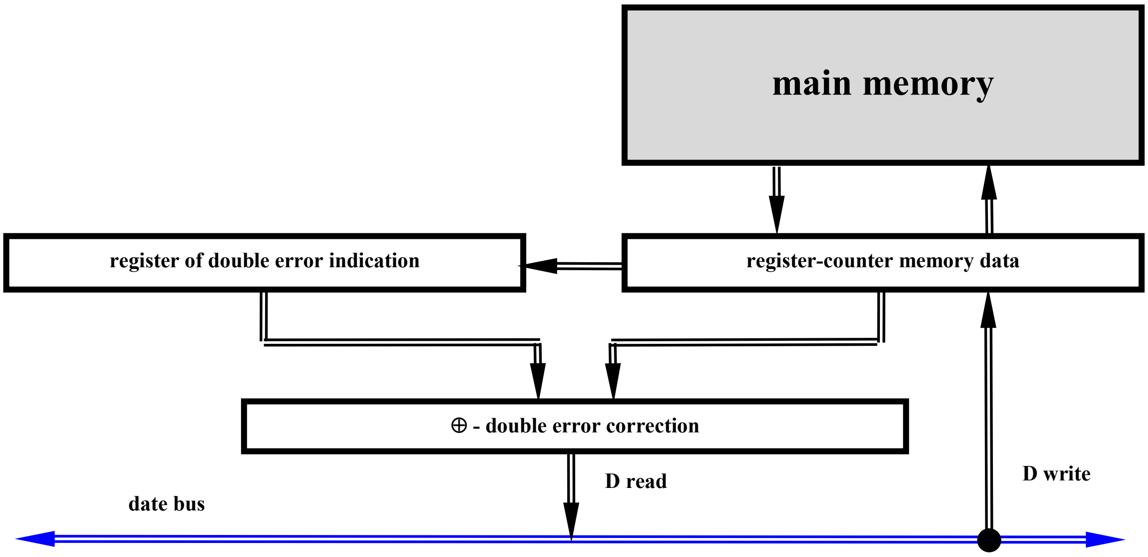
main memory
register of double error indication
register-counter memory data
⊕ - double error correction
D read
D write
date bus

Figure 13.5: Block diagram of double error repair by repeated writing and reading method

Bouble error signalization may be an initialization flag to change the location of a page or data page or segment during a virtual memory function – see Figure 13.6.

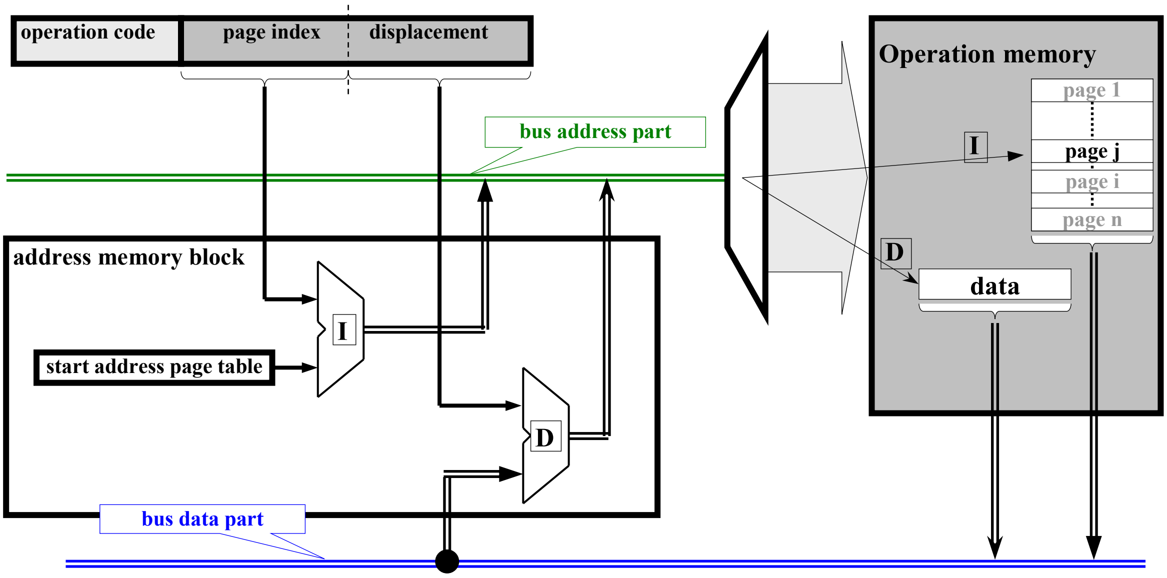

Figure 13.6: Schematic diagram of the memory paging

After repairing a single-error or double-error indication, the operating system can run the recovery routine – see Figure 13.7.

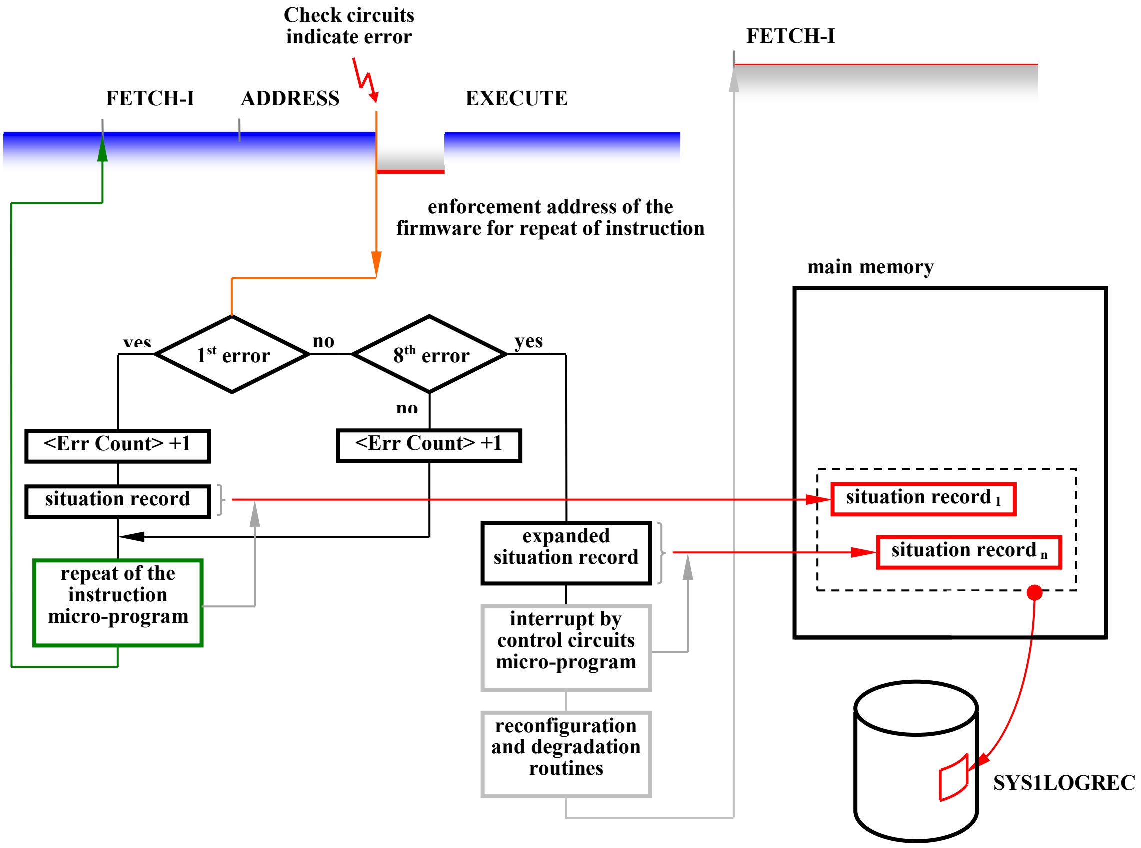
Check circuits indicate error
FETCH-I
FETCH-I
ADDRESS
EXECUTE
enforcement address of the firmware for repeat of instruction
main memory
yes
1st error
no
8th error
yes
no
<Err Count> +1
<Err Count> +1
situation record
expanded situation record
repeat of the instruction micro-program
interrupt by control circuits micro-program
reconfiguration and degradation routines
situation record 1
situation record n
SYS1LOGREC

Figure 13.7: Diagram activity of the system recovery after the error

List of figures

List of tables