CONCEPT OF PSEUDO-RING SELF-TEST OF THE RAM

Omar Saeed Al-Mushayt
MIS Department, Business College,
King Khalid University, Abha, KSA.
omushayt@kku.edu.sa

ABSTRACT: *This paper presents the concept of pseudo-ring self-test (π-test) of the random access memory (RAM), The distinctive particularity of π-testing is that RAM truly is self-tested.. The process of emulation can be controlled and analytically described. This allows adapting optimally the π-test parameters in order to obtain the maximal value of fault coverage. The elaborated methodical and software tools help to reach this aim.*

Keywords: Embedded system, RAM, Pseudo-Ring Self-Test

1. INTRODUCTION

RAM truly came about in 1966 when Robert Dennard from IBM's research center came up with the basic idea for Dynamic Random Access Memory (commonly referred to as DRAM or mostly RAM). Dennard had gone home for the day, and shortly later was somehow inspired by the basic idea for making DRAM. This turned out to be the most important advances in computer technology. This was just the beginning. It wasn't until the 1970s, that Intel released the first RAM chip called the 1103 [1].

Random access memory (RAM) is the best known form of computer memory. RAM is considered "random access" because you can access any memory cell directly if you know the row and column that intersect at that cell.

The opposite of RAM is serial access memory (SAM). SAM stores data as a series of memory cells that can only be accessed sequentially (like a cassette tape). If the data is not in the current location, each memory cell is checked until the needed data is found. SAM works very well for memory buffers, where the data is normally stored in the order in which it will be used (a good example is the texture buffer memory on a video card). RAM data, on the other hand, can be accessed in any order.

Similar to a microprocessor, a memory chip is an integrated circuit(IC) made of millions of transistors and capacitors. In the most common form of computer memory, dynamic random access memory (DRAM), a transistor and a capacitor are paired to create a memory cell, which represents a single bit of data. The capacitor holds the bit of information -- a 0 or a 1 (see How Bits and Bytes Work for information on bits). The transistor acts as a switch that lets the control circuitry on the memory chip read the capacitor or change its state.

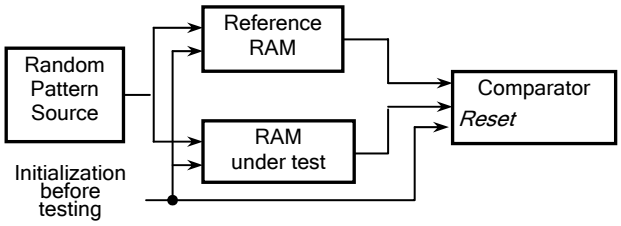

Figure 1. Scheme of the RAM random testing

It is easy to see that the samples (1) are pseudorandom sequences, generated by trivial Linear Feedback Shift Register (LFSR) described by the polynomial p(x) = 1+x+x2.

Example 1. In figure 1 is presented the process of generation of the first sequence, a, in (1).

But the pioneer and fundamental works in the random testing RAM should be considered to be the papers Scheme of random testing RAM contains the reference memory, the memory under test, and the comparator (Figure 1).

Considering scheme, shown in the figure 2, RAM test quality, i.e. fault coverage, is estimated by the length of random testing. As it is remarked in [4] the test length is a function of the fault, number of cells, the detection uncertainty, the initial state, an the pattern probabilities. All calculus are applicable for the truly random test, but not for pseudorandom testing, where tests are generated repeatable, starting by an initial seed.

In the same time, the test results from pseudorandom tests are not well suited for Built-In Self-Test (BIST) [5]. For

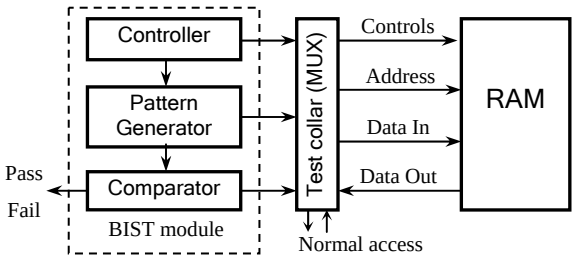

Figure 2. Typical RAM BIST architecture

implementation of the BIST RAM in [6, 7] preference is given to the deterministic test technique, based on the March algorithm. A typical RAM BIST architecture is shown in figure 2. Controller and pattern generator can be configured to execute the deterministic or the pseudorandom testing [8].

March algorithm is designed to cover a predefined set of faults. But a new RAM fault needs a novel test algorithm and, so, BIST module must be configured or/and reprogrammed for it.

Another test approach, been really a self-test scheme, was proposed recently [9, 10]. This test technique, called pseudo-ring (or π-) testing, is based on emulation of the

linear automaton, in particular, LFSR, by memory itself. In this case, no pattern generator is needed. The test quality is estimated at the end of π-test iteration. There is more degree of freedom to control the test procedure than in the deterministic or in the random testing. And such control is not so expensive.

The paper is organized as follows: section 2 introduces in the π-testing RAM; section 3 describes the hardware and software tools; section 4 considers some concluding remarks.

2. Π-TESTING CONCEPT

Notion of pseudo-ring comes from the *ring-like* testing of digital circuits [11]. In the ring-like testing circuit is reconfigured so that it is transformed to a linear (or nonlinear) automaton, i.e. LFSR. Test procedure is quite simple: automaton is clocked during a period of time *T*, equal to:

$$T = 2^m - 1, \qquad (2)$$

where *m* is the number of register stages.

After this period the final state **Fin** of register is compared with expected one. In particular, the expected state is equal to initial **Init** state. Then the comparison is made as:

$$\mathbf{Init} \gtrless \mathbf{Fin}. \qquad (3)$$

The level of confidence of the test quality is estimated by the detection uncertainty.

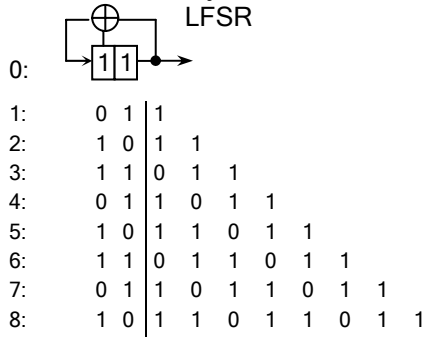

Figure 3. State-diagrams of pattern generator

Memory, at the top-level of abstraction, can be interpreted as a long register with random access to its stages. Of course, there is no reason to reconfigure so long "register" to the structure of a linear automaton. We go in another way.

The idea of *pseudo-ring* RAM testing is to use a set of memory's cells as the register stages. Second, after each clock of time are shifted not the data in the register (see figure 1), but the *virtual* register itself. The only what is needed, is to help supplementary to push from outside this *virtual automaton* in the memory address space. Complete transition of the virtual LFSR across all addresses of the memory is called the π-*test iteration*, or simple, π-*iteration*. Π-test iteration consists of: *initialization* of virtual automaton, *pushing* this automaton in the space of memory array, *unloading* the automaton final state, and *analysis* of the results.

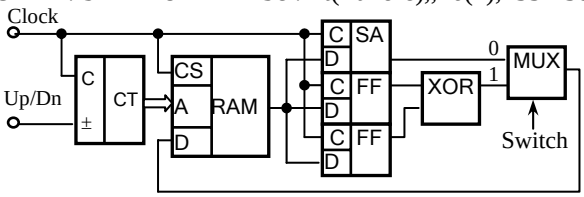

Figure 5. Scheme of π-testing RAM

Go/no go π-iteration is evaluated as in the ring-like testing: comparing at the end of π-iteration the final and expected states of the virtual automaton, i.e. register.

Consider the *Example* 2. Let the LFSR and its initial state is the same as it is shown in figure 1. Conventionally suppose

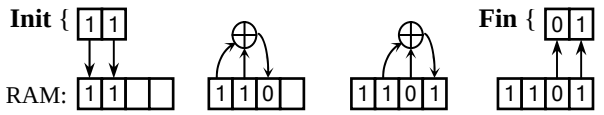

Figure 4. Π-test iteration diagram

that the memory size array is equal to 4; the cell size is equal to 1. Let the addressing mode is counted up. Π-test iteration starts with *loading* the virtual register by an initial seed. Let the first two memory cells (with address 0 and 1) play the role of the stages of virtual register. Load these stages by the initial seed <1; 1> (Figure 4, a).

In accordance with structure of polynomial $p(x)$ are performed (on corresponding cells) the read and modulo operations. Continue this operations until the final state will be reached. If don't take into account the peculiarity of performing the read-write operations, then it can be accepted that $N+m$ conditional clocks of time were carried out to move the virtual automaton in the memory cells address space. Since $N \gg m$, then the complexity of π-iteration is of order:

$$O(\pi\text{-iteration}) = N. \qquad (4)$$

In the above example was presented the main idea of the "mechanism" (algorithm) of the π-iteration, which is a constitutive part of the π-testing RAM. Generally, π-test procedure consists in the execution of the controlled π-test iterations finalized by analyzing of the results, i.e. the automaton states. In fact, there are three controlling parameters (degree of freedom):

- automaton structure, defined by polynomial $p(x)$;
- initial seed in the π-iteration;
- Addressing mode or trajectory of automaton.

Tell some words about each of these parameters. As a rule, linear automaton follows the structure of an irreducible polynomial $p(x)$ of degree $m = \deg p(x)$. Since π-iteration must be performed at least *T* clocks of time, then:

$$m \le \log_2 N, \qquad (5)$$

where *N* is the memory array size.

In dependence of the seed value at the beginning of each π-iteration, can be distinguished two types of the π-test scheme: *via-register* and *self-memory*. In the self-type

scheme the final state of the previous π-iteration is the initial state of the next π-iteration. In this case, π-test procedure is executed non-stop. In the via-scheme hardware and time overhead are needed to reload the virtual automaton in the each π-iteration.

There are three basic addressing modes for π-test executions: *count up*, *count down*, and *random*. Need to outline that in some works (for example, [12,13]) is remarked, that use of different initial conditions such as address order or background changing can increase the test quality of the March algorithm.

Another remarkable property of the π-testing, that must be denoted, is the invariability of the testing scheme. It means that the same π-test scheme can be applied (without essential modifications) as for bit-oriented so for the word-oriented memories, and as for single-port so for multi-port memories. In this context it should be use a specific method of calculation of such parameter as the test length. Since in dependence of the memory type, the π-testing scheme can has different variants of its implementation, then it is reasonable to evaluate the π-test length by number of states that the virtual automaton has gone in the predefined period of π-test time. Let k be the number of π-iteration. Considering estimation (4), the π-test length L is of order:

$$L = O(kN). \quad (6)$$

Finally, in spite of the word "random" in the notion "pseudo-random", the behavior of the (virtual) automaton is just very deterministic and predictable. Π-test approach has a fundamental mathematical support, namely, the theory of linear automaton.

3. HARDWARE AND SOFTWARE TOOLS

Memory, manufactured as a circuit unit or as an embedded block, is almost ready automaton, i.e. contains almost all components necessary to implement the π-test procedure. Test-engineer should only to select the single scheme of π-testing from the proposed one, then to specify the control parameters, to simulate the RAM under test for the selected faults, and to analyze the obtained results.

3.1. Hardware tools

In accordance with the π-test technique hardware overhead must carry out the modular operation and to push virtual automaton in the address space of memory array. Consider further an example, which is the instructive and illustrative from the practical point of view.

Example 3. Let as it was in examples 1 and 2, $p(x)=1+x+x^2$, $m=2$. Memory array size is multiple with m. Other control parameters are: *trajectory* – counting up and self-testing; *initial seed* – a degeneration (zero) state. Memory is a standard static RAM circuit. Then, it is enough to use an Up/Down counter to generate the address of the selected cells. Two flip-flops FF and a XOR gate will be used to save the read data and to calculate the sum modulo 2 of these data.

In accordance with random (see figure 2) and deterministic testing, RAM must be initialized before. In the π-testing scheme the signature analyzer is used as for initialization of RAM, so for "processing" the output data. The resulted scheme, for analyzed example, is shown in figure 5 (make comparison with figures 2 and 3).

Π-test starts with initialization of RAM by patterns generated by signature analyzer SA (output of multiplexor MUX is switch on input 0). In this iteration of initialization counter CT will up from low to high address value. Further, are executed non-stop three π-iterations, where counter follows the states: $0(rd)\rightarrow1(rd)\rightarrow1(wr)$; $1(rd)\rightarrow2(rd)\rightarrow3(wr)$ etc., *rd* and *wr* are read and write operations. In each such iteration different backgrounds will be generated. From figure 4 is resulted that backgrounds are: 1101, 1011, and 0110. On the end of iteration corresponding virtual automaton final state is expected. The expected final states and the signature analyzer states are values for π-test quality estimation.

3.2. Software tools

Test-engineer can elaborate and debug a π-test procedure by Development and Simulation Tools Environment (DSTE).

4. CONCLUSIONS

In this paper the concept, hardware and software tools of the RAM pseudo-ring (π-) self-test are presented. Test engineer has 3 degree of freedom to control the π-test procedure. Elaborated software tools allow to debug π-test algorithms, simulate faulty memory and evaluate the π-test quality.

Results of the trivial π-testing fault coverage are presented. 13 single and 15 two-cell functional faults of static RAM were simulated in this trivial test experiment. Length of π-test is equal to $4N$, where N is the size of memory array, and for its implementation is needed an up-down counter, two flip-flops and a XOR gate. Other practical π-test schemes are not more complex than the trivial one.

6. REFERENCES

[1] S. P. Tyul`kin, "Programatestirovaniya OZU", *Mikroprocessorny`esredstvaisistemy`*, 1987, no.1.

[2] P. Fosse and R. David, "Random testing of memories", *Proc. 7th Conf. Gesellschaft fur Informatik (GF'77)*, Nuremberg, Germany, Sept 1977.

[3] W. H. McAnney, P.H.Bardell, and V.P.Gupta,"Random testing for stuck-at-storage cells in an embedded logic", *Proc. Int. Test Conf.*, Philadelphia, PA, Oct 1984.

[4] A.Fuentes, R.David, and B.Courtois, "Random Testing versus Deterministic Testing of RAM's", *IEEE Trans. OnComputers*, Vol. 38, No. 5, May 1989, pp. 637-650.

[5] A. van de Goor, *Testing Semiconductor Memories*, ComTex Publishing, Gouda, The Netherlands, 1998.

[6] D. S. Suk, and S.M.Reddy, "A March Test for Functional Faults in Semiconductor Ranom Access memories", *IEEE Trans. OnComputers*, Vol. 30, May 1981.

[7] M. Marinescu, "Simple and Efficient Algorithms for functional RAM testing", *Proc. Int. Test Conf.*, 1982.

[8] C.-T. Huang, J.-R. Huang, C.-F. Wu, and T.-Y. Chang. "A Programmable BIST core for embedded DRAM", *IEEEDesign&Test of Computers*, vol. 16, No.1, 1999, pp. 59-70.

[9] G. Bodean, "PRT: Pseudo-Ring Testing – A Method for Self-Testing RAM", *IEEE-TTTC Int. Conf. On Automation, Quality and Testing, Robotics: AQTR 2002 (THETA 13)*, Tome 1, Cluj-Napoca, Romania, May 2002, pp. 295-300.

[10] G. Bodean, D.Bodean, A.Labunetz, "New Schemes for Self-Testing RAM", *Design, Automation and Test in Europe (DATE 2005)*, Munich, Germany, March 2005.

[11] I.P. Litikov "Ring-like testing of Digital circuits" *Measurement*, vol.4, No.1, 1986, pp.2-6.

[12] D. Niggemeyer, J. Otterstedt, and M. Redeker, "Detection of Non-classical Memory Faults using Degrees of Freedom in March testing", *11th Workshop Test methods and Reliability of Circuits and Systems*, Potsdam, February 1999.

[13] B. Sokol, and V. N. Yarmolik, "Memory Faults Detection Techniques With Use of Degrees of Freedom in March Tests", *Proc. IEEE East-West Design & Test Workshop (EWDTW'05)*,Odessa, Ukraine, Sept 2005.