An Open Notation for Memory Tests

Ad J. van de Goor, Aad Offerman, and Ivo Schanstra: Towards a Uniform Notation for Memory Tests (presented at the European Design & Test Conference and Exhibition 96)
Aad Offerman and Ad J. van de Goor: An Open Notation for Memory Tests (presented at the IEEE International Workshop on Memory Technology, Design and Testing, 1997)
There is a font issue here. I have emailed Adrian Offerman to find out if he knows of a font that contains the symbols used here. He sent me his xbm bitmaps, but they are in the Symbol and Wingdings fonts in Word. The only remaining issue is how to make them display in HyperTerminal.

Memory Test Verification and Generation

A. Offerman, Automatic Verification and Generation of Memory Tests (Masters thesis, August 1995)
While it is certainly possible to program this piece, it would not be appropriate to put this into an embedded environment. At best, this would be an off-line activity.

Simplified notation for terminals

Terminal font has single arrows
↕ up/down = 0x12 = ^R
↓ down = 0x19 = ^Y
↑ up = 0x18 = ^X

Standard test algorithms

S. Hamdioui, R. Wadsworth, J.D. Reyes, and A.J. van de Goor, Memory Fault Modeling Trends: A Case of Study (Journal of Elecronic Testing, Thoery and Application JETTA, Vol. 20, pp. 245-255, 2004)
Alexander Paalvast, Testing Single Inline Memory Modules (SIMMs) Theory and practice (Masters thesis, December 1999)
Trade off complexity with fault detection. On-going maintenance of test notation parser determines what is possible.

Table 1. ***Examples of base tests (BTs).***

No. BT name	Test length	Description
1 SCAN [1]	4n	{⇑ (*w*0);⇑ (*r* 0);⇑ (*w*1);⇑ (*r*1)}
2 MATS+ [16]	5n	{⇕(*w*0);⇑ (*r*0,*w*1);⇓ (*r*1,*w*0)}
3 MATS++ [6]	6n	{⇕ (*w*0);⇑ (*r*0,*w*1);⇓ (*r*1,*w*0, *r*0)}
4 March C– [14, 18]	10n	{⇕ (*w*0);⇑ (*r*0,*w*1);⇑ (*r*1,*w*0);⇓ (*r*0,*w*1);⇓ (*r*1,*w*0); ⇕ (*r*0)}
5 PMOVI [8]	13n	{⇓ (*w*0);⇑ (*r*0,*w*1, *r* 1);⇑ (*r*1,*w*0, *r* 0); ⇓ (*r*0,*w*1, *r* 1);⇓ (*r*1,*w*0, *r*0)}
6 March SR [9]	14n	{⇓ (*w*0);⇑ (*r*0,*w*1, *r*1,*w*0);⇑ (*r*0, *r* 0); ⇑ (*w*1);⇓ (*r*1,*w*0, *r*0,*w*1);⇓ (*r*1, *r*1)}
7 March SS [11]	22n	{⇕ (*w*0);⇑ (*r*0, *r*0,*w*0, *r*0,*w*1);⇑ (*r*1, *r*1,*w*1, *r*1,*w*0); ⇓ (*r*0, *r*0,*w*0, *r*0,*w*1);⇓ (*r*1, *r*1,*w*1, *r*1,*w*0); ⇕ (*r*0)}
8 March G [17]	23n	{⇕ (*w*0);⇑ (*r*0,*w*1, *r*1,*w*0, *r*0,*w*1);⇑ (*r*1,*w*0,*w*1); ⇓ (*r*1,*w*0,*w*1,*w*0);⇓ (*r*0,*w*1,*w*0);⇑ (*r*0,*w*1, *r* 1);⇑ (*r*1,*w*0, *r*0)}
9 March RAW [10]	26n	{⇕ (*w*0);⇑ (*r*0,*w*0, *r*0, *r*0,*w*1, *r* 1);⇑ (*r*1,*w*1, *r*1, *r*1,*w*0, *r* 0); ⇓ (*r*0,*w*0, *r*0, *r*0,*w*1, *r* 1);⇓ (*r*1,*w*1, *r*1, *r*1,*w*0, *r* 0); ⇕ (*r*0)}
10 Hammer [19]	49n	{⇑ (*w*0);⇑ (*r*0, 10 ∗ *w*1, *r* 1);⇑ (*r*1, 10 ∗ *w*0, *r* 0);

		⇓ (*r*0, 10 ∗ *w*1, *r* 1);⇓ (*r*1, 10 ∗ *w*0, *r*0)}
11 GalColumn	6n+4nR	{⇑ (*w*0);⇑*b* (*w*1*b*, *col*(*r*0, *r*1*b*),*w*0*b*);
		⇑ (*w*1);⇑*b* (*w*0*b*, *col*(*r*1, *r*0*b*),*w*1*b*)}
12 GalRow	6n+4nC	{⇑ (*w*0);⇑*b* (*w*1*b*, *row*(*r*0, *r*1*b*),*w*0*b*);
		⇑ (*w*1);⇑*b* (*w*0*b*, *row*(*r*1, *r*0*b*),*w*1*b*)}
13 WalkColumn	8n+2nR	{⇑ (*w*0);⇑*b* (*w*1*b*, *col*(*r*0), *r*1*b*,*w*0*b*);
		⇑ (*w*1);⇑*b* (*w*0*b*, *col*(*r*1), *r*1*b*,*w*0*b*)}
14 WalkRow	8n+2nC	{⇑ (*w*0);⇑*b*(*w*1*b*, *row*(*r*0), *r*1*b*,*w*0*b*);
		⇑ (*w*1);⇑*b*(*w*0*b*, *row*(*r*1), *r*1*b*,*w*0*b*)}

[1] M.S. Abadir and J.K. Reghbati, “Functional Testing of Semiconductor Random Access Memories,” ACM Computer Surveys, vol. 15, no. 3, pp. 175–198, 1983.
[6] M.A. Breuer and A.D. Friedman, Diagnosis and Reliable Design of Digital Systems, Woodland Hills, CA, USA: Computer Science Press, 1976.
[8] J.H. De Jonge and A.J. Smeulders, “Moving Inversions Test Pattern is Thorough,Yet Speedy,” Comp. Design, 1976, pp. 169–173.
[9] S. Hamdioui and A.J. van de Goor, “Experimental Analysis of Spot Defects in SRAMs: Realistic Fault Models and Tests,” Proc. of Ninth Asian Test Symposium, 2000, pp. 131–138.
[10] S. Hamdioui, Z. Al-Ars, and A.J. van de Goor, “Testing Static and Dynamic Faults in Random Access Memories,” Proc. of IEEE VLSI Test Symposium, 2002, pp. 395–400.
[11] S. Hamdioui, A.J. van de Goor, and M. Rodgers, “ March SS: A Test for All Static Simple RAM Faults,” Proc. IEEE InternationalWorkshopon Memory Technology, Design, and Testing, 2002, pp. 95–100.
[14] M. Marinescu, “Simple and Efficient Algorithms for Functional RAM Testing,” in Proc. of International Test Conference, 1982, pp. 236–239.
[16] R. Nair, “An Optimal Algorithm for Testing Stuck-at Faults Random Access Memories,” IEEE Trans. on Comp., vol. C-28, no. 3, pp. 258–261, 1979.
[17] D.S. Suk and S.M. Reddy, “A March Test for Functional Faults in Semiconductors Random-Access Memories,” IEEE Trans. on Comp., vol. C-30, no. 12, pp. 982–985, 1981.
[18] A.J. van de Goor, Testing Semiconductor Memories, Theory and Practice, ComTex Publishing, Gouda, The Netherlands, 1998.
[19] A.J. van de Goor and J. de Neef, “Industrial Evaluation of DRAMs Tests,” Proc. of Design Automation and Test in Europe, 1999, pp. 623–630.

Additional references

A Memory Debug Methodology Using BIST
A Microcode-based Memory BIST Implementing Modified March Algorithm
A Programmable Data Background Generator for March Based Memory Testing
A Systematic Method for Modifying March Tests for Bit-Oriented Memories into Tests for Word-Oriented Memories
DETECTING FAULTS IN THE PERIPHERAL CIRCUITS AND AN EVALUATION OF SRAM TESTS
Detecting Intra-Word Faults in Word-Oriented Memories
Dynamic Faults in Random-Access-Memories: Concept, Fault Models and Tests
EMBEDDED MEMORY BIST FOR SYSTEMS-ON-A-CHIP
Evaluating Tests for Input Stuck-at Faults in Word-Oriented Static Random-Access Memories
Memory Test
Lecture 15 Memory Test
Lecture 16 Pattern Sensitive and Electrical Memory Test
Memory Test1

RAM Fault Models & Test Algorithms
Memory Testing
Testing Flash Memories
Models and Test Procedures for Flash Memory Disturbances
RAM Testing Algorithms for Detection Multiple Linked Faults
Simulation-Based Test Algorithm Generation and Port Scheduling for MultiPort Memories
RAMSES: A Fast Memory Fault Simulator
System-on-a-Chip Design and Test: Part 1 - Methods
Testing Embedded Memories in Telecommunication Systems
Testing Word-Oriented & Multi-Port Memories

Memory Test Language

Ignore white space

Tabs and spaces

Ignore comments

Prefixed with #

Increment line number

Newline

Initial

Begin test algorithm

ALLPORTS or allports or {

Port commands

Read port

r

Write port

w

Read/Write port

x

Don’t care port

-

Rest

.

Test Algorithm

Addressing order

DOWN or down or ⇩(xC8->x19)

UP or up or ⇧(xC9->x18)

UPDOWN or updown or ⇳(xCA->x12)

SOUTH or south or ↓(xCB)

NORTH or north or ↑(xCC)

NORTHSOUTH or northsouth or ↕(xCD)

EAST or east or →(xCE)

WEST or west or ←(xCF)

EASTWEST or eastwest or ↔(xD0)

Operators

+

-

Numbers

0-9

No operation

NOP or nop

End test algorithm

}

Rest

. or A-Z or a-z