Tommi Norhamo

MEMORY TESTING OF AN EMBEDDED DEVICE

Technology and Communication
2020

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Tommi Norhamo
Opinnäytetyön nimi	Memory Testing of an Embedded Device
Vuosi	2020
Kieli	englanti
Sivumäärä	48 + 1 liite
Ohjaaja	Jani Ahvonen

Tämä opinnäytetyö tehtiin ABB Oy Distribution Solutions yksikölle. Tämän opinnäytetyön tarkoituksena oli tutkia ja kehittää keinoja parantaa sulautetun laitteen itsevalvontaa. Työ helpottaisi laatuosaston työtä asiakaspalautustuotteiden muistivikojen diagnosoimisessa. Työ suoritettiin syksyn 2019 aikana.

Opinnäytetyössä kehitetään keskusmuistia testaava sulautettu ohjelmisto. Työssä käsitellään elektromagneettisia häiriöitä, erilaisten muistien rakenteita ja testaustekniikoita.

Tietoa kerättiin pääasiassa kirjallisuudesta, mutta myös tieteellisistä julkaisuista, artikkeleista ja laitteiden valmistajien teknisistä tiedotteista.

Testiohjelmisto tekee tehtävänsä eli löytää muistiviat sekä raportoi tulokset näytölle ja tiedostojärjestelmään.

Avainsanat itsetestaus, ABB, muistiviat, sulautettu

VAASAN AMMATTIKORKEAKOULU
VAASA UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Tommi Norhamo
Title	Memory Testing of an Embedded Device
Year	2020
Language	English
Pages	48 + 1 Appendix
Name of Supervisor	Jani Ahvonen

This thesis was written at ABB Oy, Distribution Solutions unit. The objective of this thesis was to explore and implement ways to improve self-testing properties of an embedded device. The aim is to help the Product Quality department to diagnose memory faults in customer returned products. The thesis was carried out during fall 2019.

The purpose of this thesis was to implement embedded software for testing memory. This thesis covers different memory architectures and memory testing fault models.

Information was mainly collected from literature, but scientific publications and papers are also used alongside device manufacturer datasheets.

The software does what it is designed for – it finds memory faults, reports them by writing them to a display and the filesystem.

Keywords Self-testing, ABB, memory faults, embedded

CONTENTS

LIST OF FIGURES AND TABLE

LIST OF APPENDICES

GLOSSARY

PCB	Printed Circuit Board
LCD	Liquid Crystal Display
CPU	Central Processing Unit
IC	Integrated Circuit
IDE	Integrated Development Environment
FPGA	Field-Programmable Gate Array
RAM	Random-Access Memory
NVRAM	Non-Volatile Random-Access Memory
DRAM	Dynamic Random-Access Memory
SDRAM	Synchronous Random-Access Memory
SRAM	Static Random-Access Memory
ROM	Read-Only Memory
NVM	Non-Volatile Memory
JTAG	Joint Test Action Group
I/O	Input/Output
MC	Memory Cell
FS	File System
FG	Field Gate
CG	Control Gate

opcode	Operation Code
RAS	Row-Address Strobe
CAS	Column-Address Strobe
CE	Chip Enable
OE	Output Enable
WE	Write Enable
CL	CAS Latency
WL	Write Latency
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
SA	Sense Amplifier
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
Dual-Port RAM	DPRAM
Hypertext Markup Language	HTML
SER	Soft Error Rate
SEU	Single-Event-Upset
Trench with Internal Charge	TIC
Trench with External Charge	TEC
Stacked Capacitor	SC
Error Correcting Code	ECC

1 INTRODUCTION

This thesis was made for ABB Oy, Distribution Solutions unit. It is made to improve the self-testing properties of an embedded device-

Some of the embedded devices have been returned to the manufacturer due to different faults, such as Liquid Crystal Display (LCD), bus, memory, real-time clock, transistor or other small component faults. This thesis is about memory testing to spot defective Random-Access Memory (RAM) chips.

The microprocessors used in the embedded devices (MPC8247) do not have enough internal RAM for normal operation, so external RAM must be supplied. There are two different RAM chips attached to the Central Processing Unit (CPU) module that is being tested and the goal is to find which RAM chip causes the error and the error bits and their memory address. With this information it would be possible to prove if the memory Integrated Circuit (IC) vendor sells defective chips or if the memory faults are caused in the production.

2 COMPUTER MEMORY

In computer sciences memory refers to the hardware IC that store information for immediate use. Memory types are divided into two categories, volatile and non-volatile. Understanding the differences between memory types creates basic understanding of computer memory used later in this document. /1/

2.1 Non-volatile Memories

Non-volatile memory (NVM) retrieves information after it has been power cycled, whereas volatile memory does not. Examples of NVM are flash memory, Read-Only Memory (ROM), Non-Volatile Random-Access Memory (NVRAM), hard disks, floppy disks and optical disks. /1/

Flash memory is an electronic non-volatile computer memory. Writing to flash memory is done in pages and the page size depends on the flash memory architecture and manufacturer. Before flash memory can be written, it must be first erased, and it is done in sectors (or blocks). The downside of flash memory is its limited write cycles of 1,000 to 100,000 times. /1/

Flash memories typically store information in memory cells made from floating gate transistors. Two tin oxide layer separated transistors are knowns as a Floating Gate (FG) and Control Gate (CG). FG's link to word line is through the control gate. If the link is in place, the cells value is 1. See Figure 1 for illustration of a flash memory cell. To erase data stored in flash, elevated voltage is used to discharge the floating gate, by Fowler-Nordheim tunneling (Quantum-mechanical phenomenon in which an electron with insufficient energy to overcome the barrier of the gate insulator can appear on the other side). Reading and writing the cell depends on the flash memory types; NAND and NOR. /3/

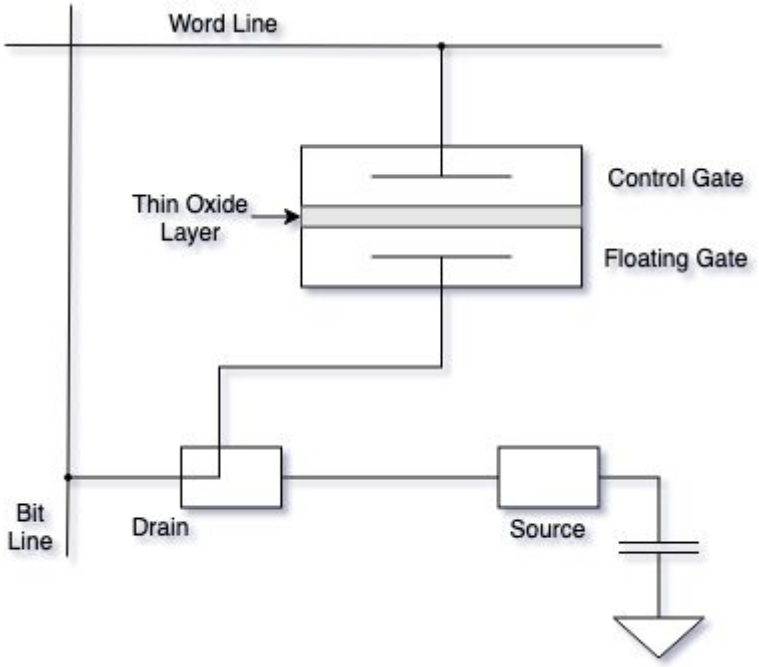


Figure 1. A flash memory cell. /1/

In NOR flash, each end of the memory cell is connected to the source line and the other end directly to a bit line resembling a NOR gate (see Figure 2). The storage transistors are in parallel, so reading out is done by gating off all expect the selected transistor. The flash cell is reset with the use of Fowler-Nordheim tunneling. NOR-type flash allows a single word to be written or read from location. NOR flash is used in embedded applications because of its low read latencies and because it allows for both code execution and data storage in a single memory product. /3/

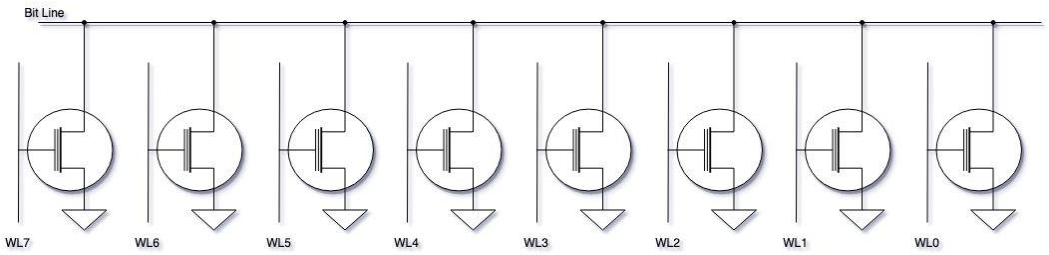


Figure 2. NOR flash structure. /3/

In NAND flash the memory cells are connected in series to form a NAND gate (see Figure 3). Transistors are connected in series and reading out is done by holding HIGH all gates but the selected transistor. This selected transistor then resolves the

conduction of the series string based on the charge on its floating gate. To write flash, NAND flash uses tunnel injection method and for erasing tunnel release is used. NAND flash memory must be erased, written and read in pages. /2/ NAND flash is typically deployed in USB flash drives, memory cards and solid-state drives. /3/

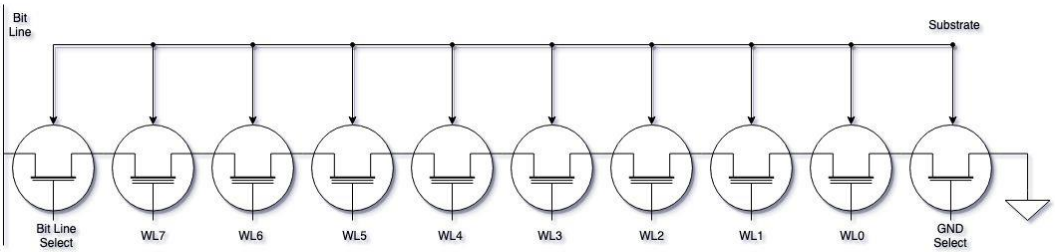


Figure 3. NAND flash structure. /3/

2.2 Volatile Memories

Computer systems expect a random-access memory out of which they operate. In modern software the code and data reside in the same place. Instruction and data requests go to this place, the memory. At any random time, any data could be needed by the microprocessor. The program code does not have to be next to the data that it manipulates. Code or more specifically instructions executed sequentially do not have to be next to each other in the memory either. This is how the name Random-Access Memory was originated. Modern volatile memories are Dynamic Random-Access Memory (DRAM) and Static Random-Access Memory (SRAM). /1/

2.2.1 SRAM

SRAM is typically less common nowadays due to its cost. It is mainly used in microprocessor cache because of its superior performance over other memory structures like DRAM. SRAM also uses the same fabrication process as the microprocessors core making the integration of cache onto the processor die less complex. /1/

A memory cell (MC) is implemented as two cross-coupled inverters accessed using two pass transistors. The cross-coupled connection creates an infinite loop that allows single bit of data to be stored indefinitely. Asserting a word line and detecting the differential voltage between the bit line pair is used to read the bit. To write a bit, the bit line must be driven with differential voltage from an external source to write data onto the memory cell. See Figure 4 below for 6 transistor SRAM cell. /1/

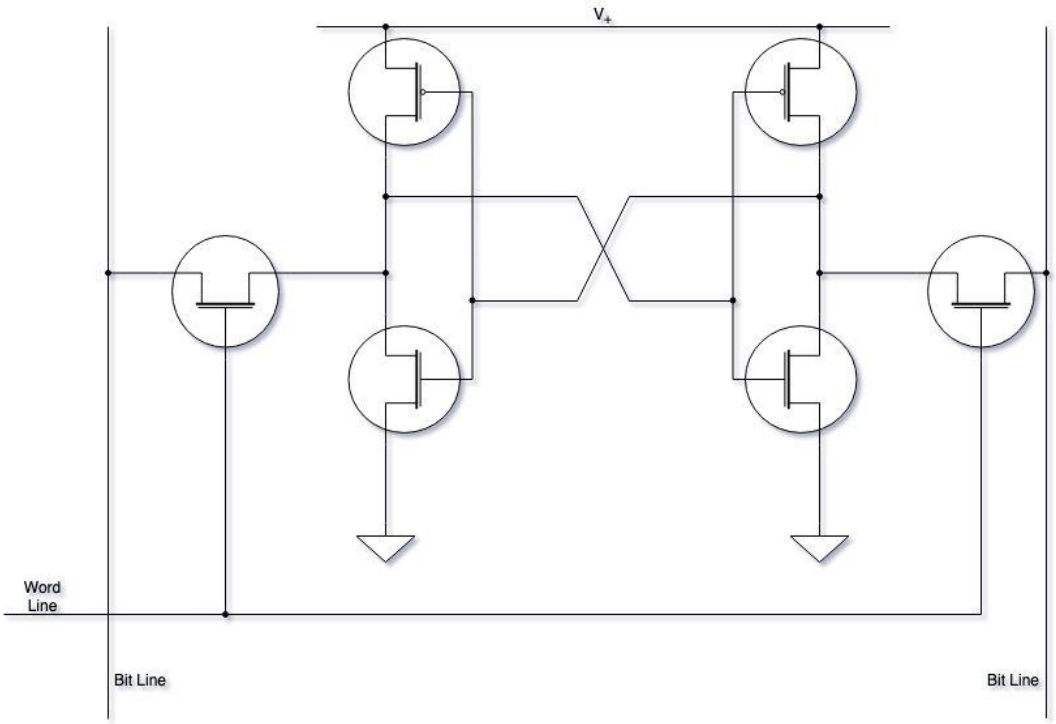


Figure 4. Basic six-transistor SRAM cell. /3/

Address decoding is a process of receiving address and giving signals to initiate and complete the operation on the addressed location. It involves feeding an address value into a binary decoder and using the output to activate the correct word line of address's memory cells. /1/

Memory arrays (such as SRAM and DRAM) usually employ two-dimensional decoding, where a part of the address accesses a single row of the array and the other part is used to select a fraction of all the columns within accessed row. /1/

Traditionally SRAM did not need clocking input, it was asynchronous. To read or write data only address, data and control signals with proper timing needed to be

applied. Reading data is a simple process (see Figure 5): the address, Chip Enable (CE'), and Output Enable (OE') are asserted. The single quotation mark and overline indicate active LOW signal. This makes the data appear on the data lines after specified time. Writing a word (see Figure 6) is done in a similar manner: address, data and CE are asserted. After specified time Write Enable (WE') pulse is sent. This results in valid data latched into memory at end of the WE' pulse. Asynchronous SRAM is a good choice for microprocessors internal cache. /3/

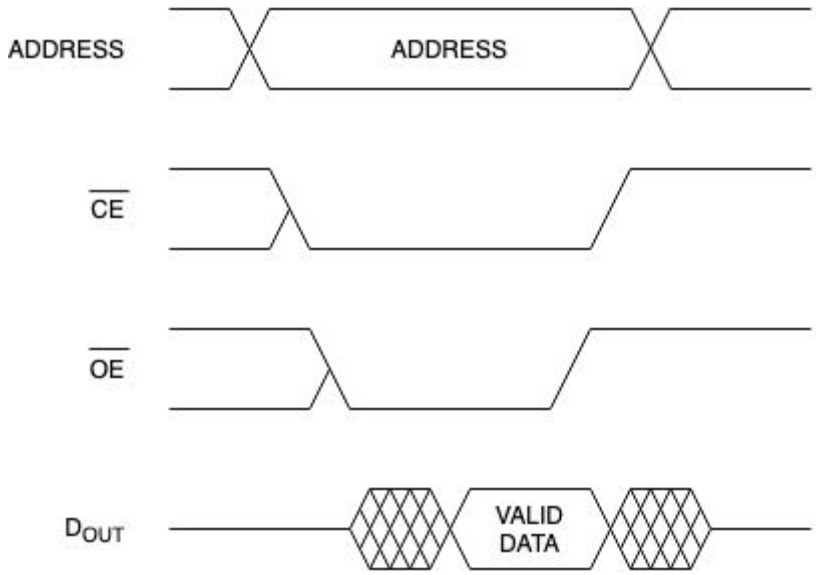


Figure 5. Asynchronous SRAM read cycle. /3/

The clocked version of SRAM is called synchronous SRAM. Because of the clocking, the speed of the synchronous SRAM is given as a maximum frequency. Synchronous SRAM is a lot more complicated than its asynchronous version. It supports double-data transfer using LOW and HIGH clock edges, bursting and interleaving data. It is an ideal candidate for microprocessors external cache. See Figure 7 for Synchronous SRAM timing with double data-rate (DDR-2) clocking during read and write operations. /3/

2.2.2 DRAM

DRAM is the "computer memory", because it's the cheapest and takes least amount of physical space. The DRAM device is typically connected to the CPU indirectly through a memory controller. Because DRAM is almost always guaranteed to be

an external device, there are some issues that a designer must consider: pins (their capacitance and inductance), signalling, signal integrity, packaging, clocking and synchronization, timing conventions. Omitting these concerns results in suboptimal or possibly non-functional design. /1/

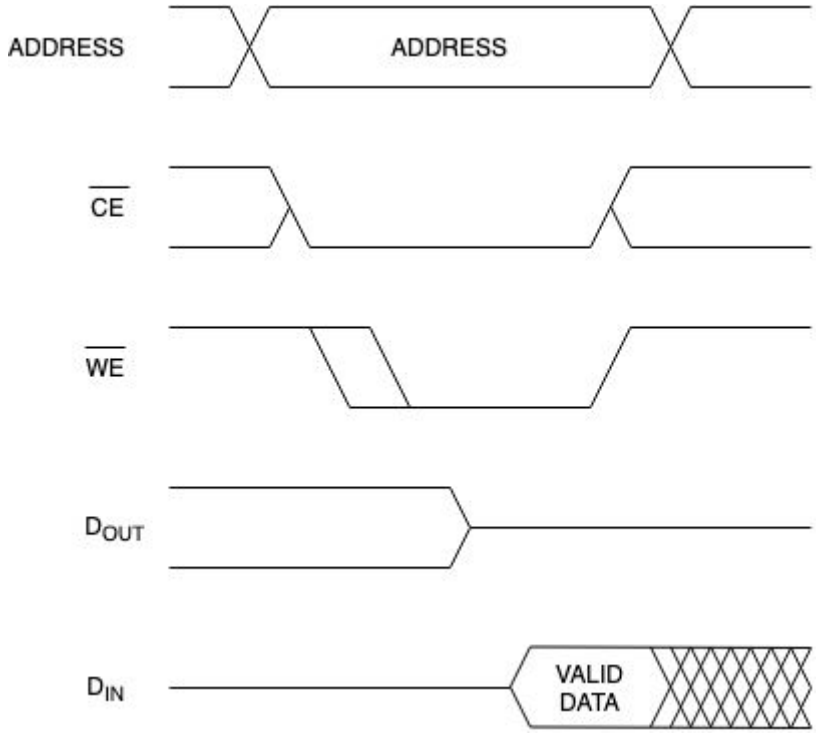


Figure 6. Asynchronous SRAM write cycle. /3/

DRAM uses a single transistor-capacitor pair for each bit (see Figure 8). This memory type is dynamic, because capacitors are used to store electrons and the capacitors leakage requires each capacitor in the memory array to be periodically refreshed. DRAM die contains multiple memory arrays, rectangular grids of storage cells with each cell holding one bit of data. A memory controller is used to access (read or write) a storage cell inside a DRAM chip by specifying a row address and a column address to the DRAM. /1/

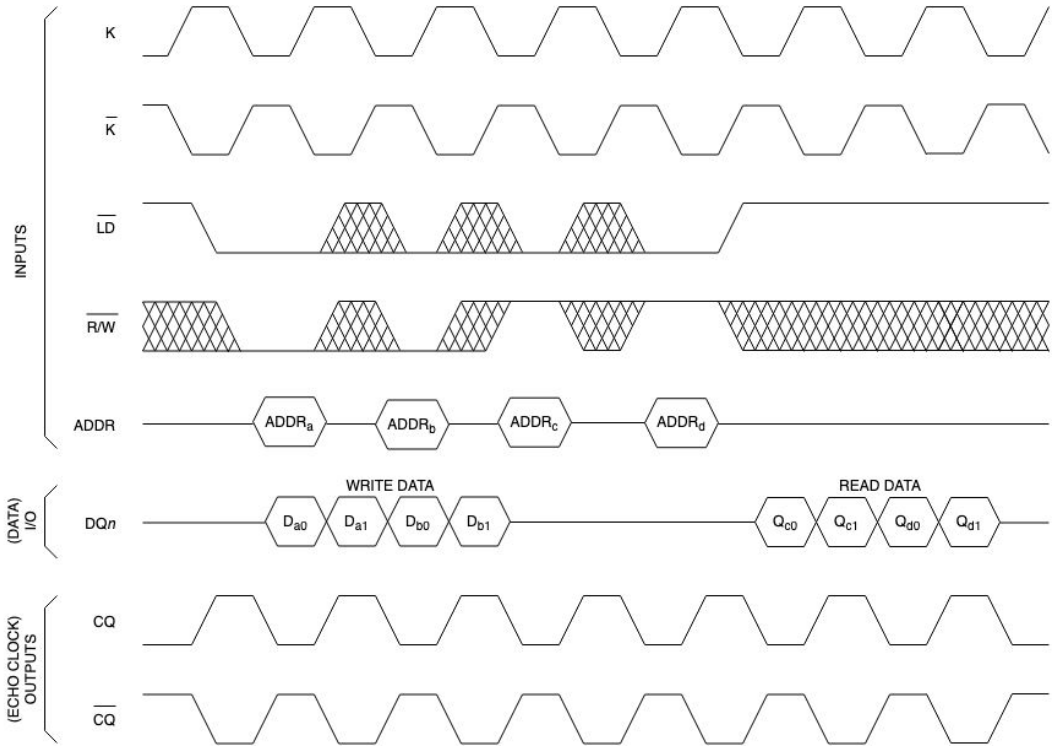


Figure 7. Synchronous SRAM read and write operations. /3/

Memory arrays within the memory chip can operate in different ways for example in unison, independently or combination of the previous ways. When the memory arrays operate as unison, they act as a unit and the memory chip usually transmits or receives a number of bits equal to the number of arrays every time the memory controller accesses the memory. To give a simple example, a x32 DRAM indicates that the DRAM has at least 32 memory arrays and column width is 32 bits. Those 32 arrays would each read 1 data bit in unison and the x32 DRAM sends out 32 data bits every time the memory controller makes a column read request. Each independently operating memory array group is called a bank. Banks are independent of each other; they can be activated, precharged and read at the same time other banks are being activated, precharged, read out, etc. See Figure 9 for DRAM array. /1/

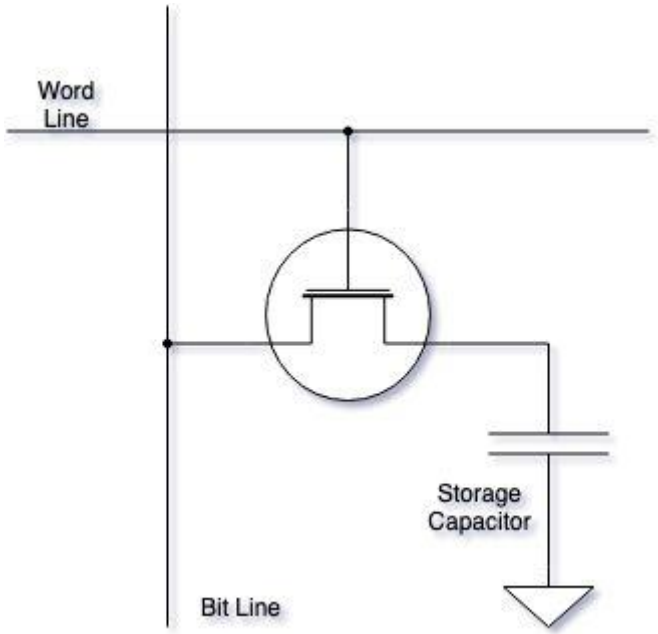


Figure 8. Basic one-transistor, one-capacitor DRAM cell. /3/

Asynchronous DRAM performs a read cycle by first asserting Row-Address Strobe (RAS') multiplexed address lines with two high-order address bits. This causes the bits to latch, setting the row gates HIGH, which turns on the Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) transmission gates, which results in coupling the respective capacitors to the column Sense Amplifier (SA'). The sense amplifiers in each column latch into the state of the bits that were held in the cells of the selected row and refresh those cells by asserting their latched level back onto the respective cell capacitors. Next, the two low-order address bits get carried by address lines and those address bits are latched by the Column-Address Strobe (CAS'). The column multiplexer selects the output from the selected SA. Because WE' is disasserted, this output is asserted onto the I/O data line. As this is asynchronous memory, the data appears on the data lines in a guaranteed time. There is no master clock in asynchronous DRAM. See Figure 10 for asynchronous DRAM timing. /3/

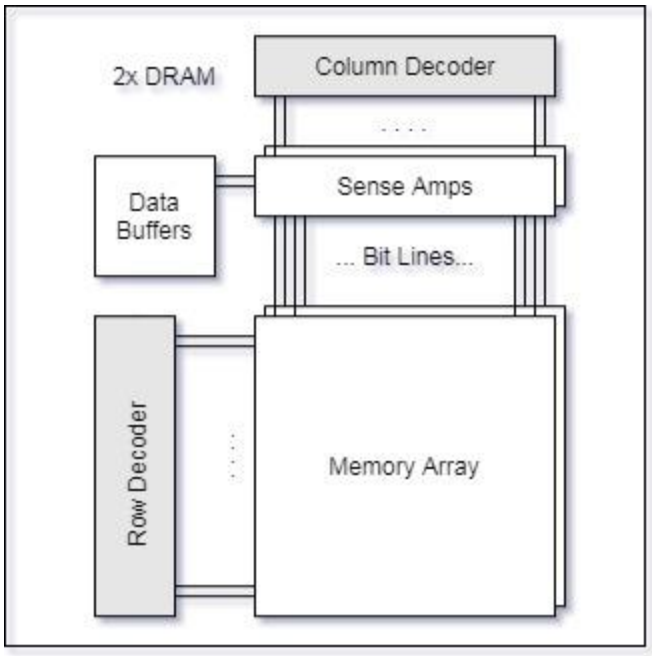


Figure 9. 2x DRAM array. /1/

Synchronous DRAM (SDRAM) is the master clocked brother of asynchronous DRAM. The clock transition in synchronous devices is used to load the row and columns addresses and to clock data in and out (read and write operations). SDRAM is typically operated in burst mode; multiple data words from consecutive column addresses clocked out in sequence. SDRAM and its asynchronous brother use the same input signal bits, RAS', CAS' and WE', which determine what happens on the next clocking edge. There is also several clock cycle delay from the assertion of the column address to the corresponding data, also called CL (CAS latency). The chip needs to know the used CL, it must be sent on a clock cycle RAS', CAS', WE' held LOW and the defined mode on the address lines. Some SDRAM varieties also have a clock latency during write cycle, also known as WL (Write Latency). WL occurs after the column address is clocked in. See Figure 11 for illustration of SDRAM which uses an external interface whose operations are synchronized by an externally applied clock. /3/

DRAM has other commands in addition of read and write: activate, refresh and precharge. The activate command is the row access command, its purpose is to move data from the cells in the DRAM array to the sense amplifiers and then restore the data back into the DRAM array cells. The time it takes for the activate command

to move data from the cell arrays to the sense amplifiers is the Row Column Delay, t_{RCD}. After the time t_{RCD} from the assertion of the row access command, the row of activated data is held in the sense amplifiers. Subsequent column read/write commands can then move data between memory controller and the sense amplifiers. Next the data has to be restored back to the DRAM cells. t_{RAS}, Row Access Strobe latency, is the time it takes for an activate command to discharge and restore the data in the memory cells. After the t_{RAS} from the assertion of the active command, the sense amplifiers have completed data restoration and they can then precharge for another activate to a different row in the same bank. /1/

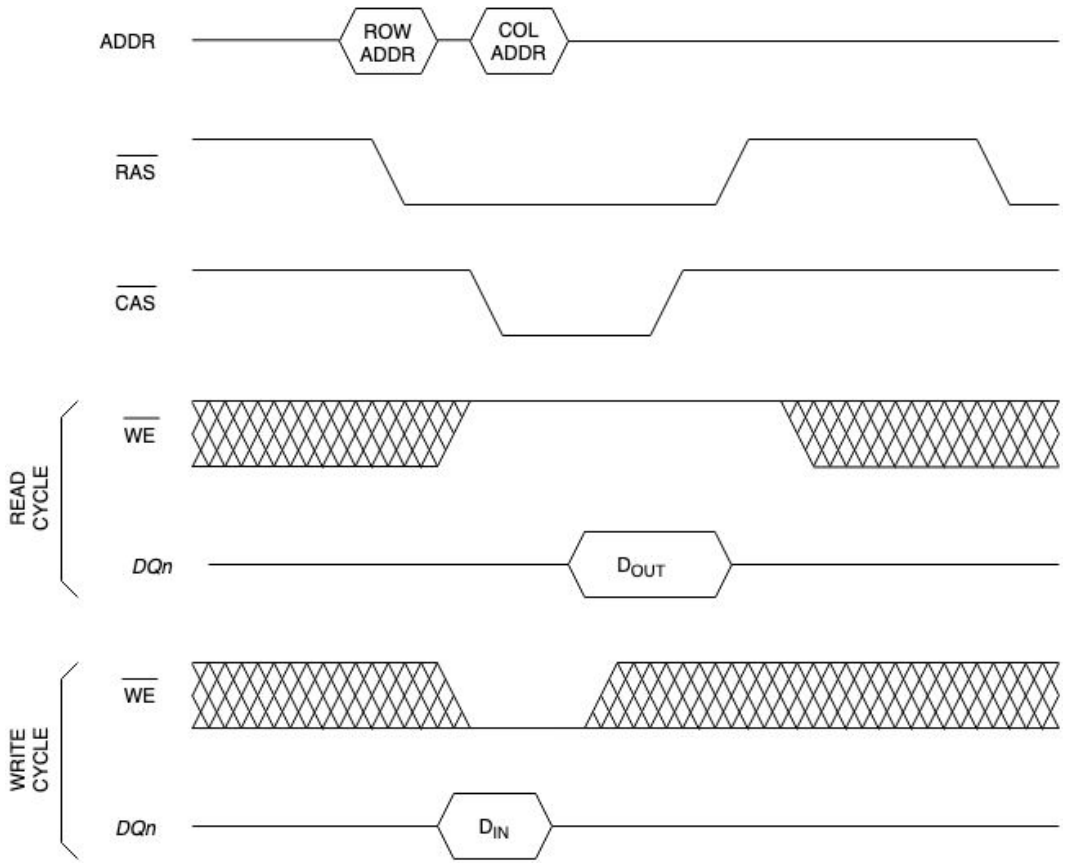
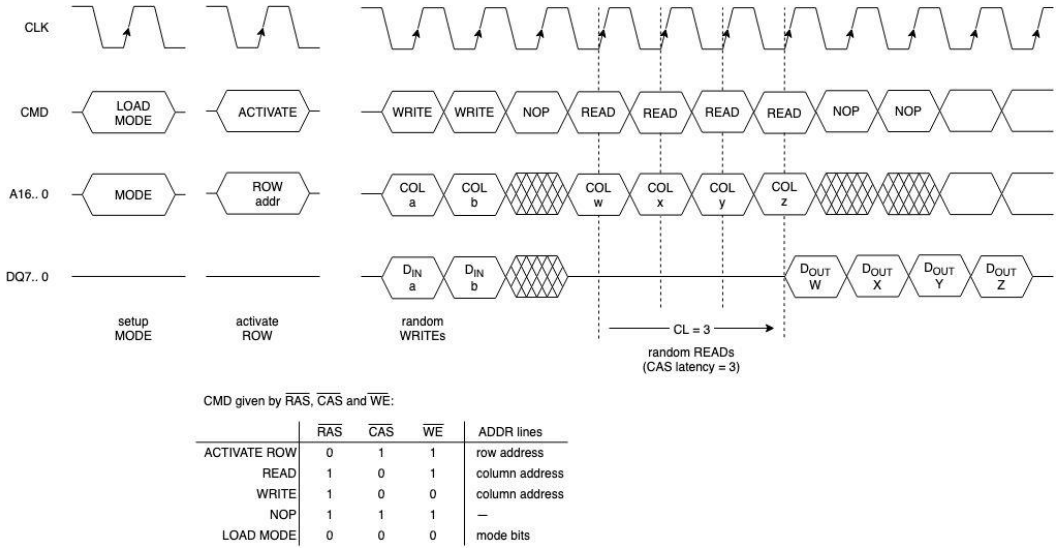


Figure 10. Asynchronous DRAM timing. /3/

Data access in DRAM consist of a two-step process. An activate command moves data from the DRAM array to the sense amplifier array. After a data row is moved into the sense amplifiers, that data is cached by the sense amplifiers for subsequent column access commands to move the data between the DRAM and the memory

controller. The precharge command completes the sequence of row access as it reset the bit lines and the sense amplifiers and prepares them for the next row access. /1/



CMD given by $\overline{RAS}$, $\overline{CAS}$ and $\overline{WE}$:

	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	ADDR lines
ACTIVATE ROW	0	1	1	row address
READ	1	0	1	column address
WRITE	1	0	0	column address
NOP	1	1	1	–
LOAD MODE	0	0	0	mode bits

Figure 11. SDRAM timing. RAS', CAS', WE' used as a 3-bit command. /3/

As DRAM is dynamic, the electrical charge stored in the storage capacitors, the memory cells, will leak out through the transistors. To main data integrity, the memory cells have to periodically read out and restored to full voltage level before the stored electrical charges decay. The refresh command handles maintained the data integrity. As long as the time interval between the refresh commands made to a row of DRAM array is shorter than the worst-case data decay time, data integrity is ensured. More frequency refreshing consumes more power and available bandwidth. Most DRAM devices use a refresh row address register to keep track of the address of the previously refreshed row. Usually it is the Memory Controller that sends a single refresh command to the DRAM device, and the DRAM device increments the address in the refresh row address register and refreshes the selected row in all the memory banks. The refresh command consists of other commands in the specified order: activate, precharge, activate and precharge. /1/

Errors in DRAM devices have been a concern for a long time. A memory error is an event that leads to the logical state of one or multiple bits being read differently from how they were last written. Memory errors can be caused by electrical or magnetic interference, by problems with the hardware or caused by the result of

corruption along the data path between the memory and the processor. The consequence of a memory error depends on the systems. Systems without support for error correction and detection, a memory error can lead to applications using corrupted data or a machine crash. Employing Error Correcting Code (ECC) allows the detection and correction of one or multiple bit errors. Uncorrectable errors, such as too many affected bits, can lead to a forced machine shutdown. /17/

2.3 Memory Controller

In today's computer systems processors and I/O devices access data in the memory using memory controller. The memory controller is a digital circuit that manages the flow of data between the microprocessor and external memory circuit. It contains the logic to read and write data to DRAM and refresh the data at set intervals. See Figure 12 for illustration of memory controller. The memory controller handles reading and writing by selecting row and column data addresses of the DRAM as the inputs to the multiplexer circuit. /1/

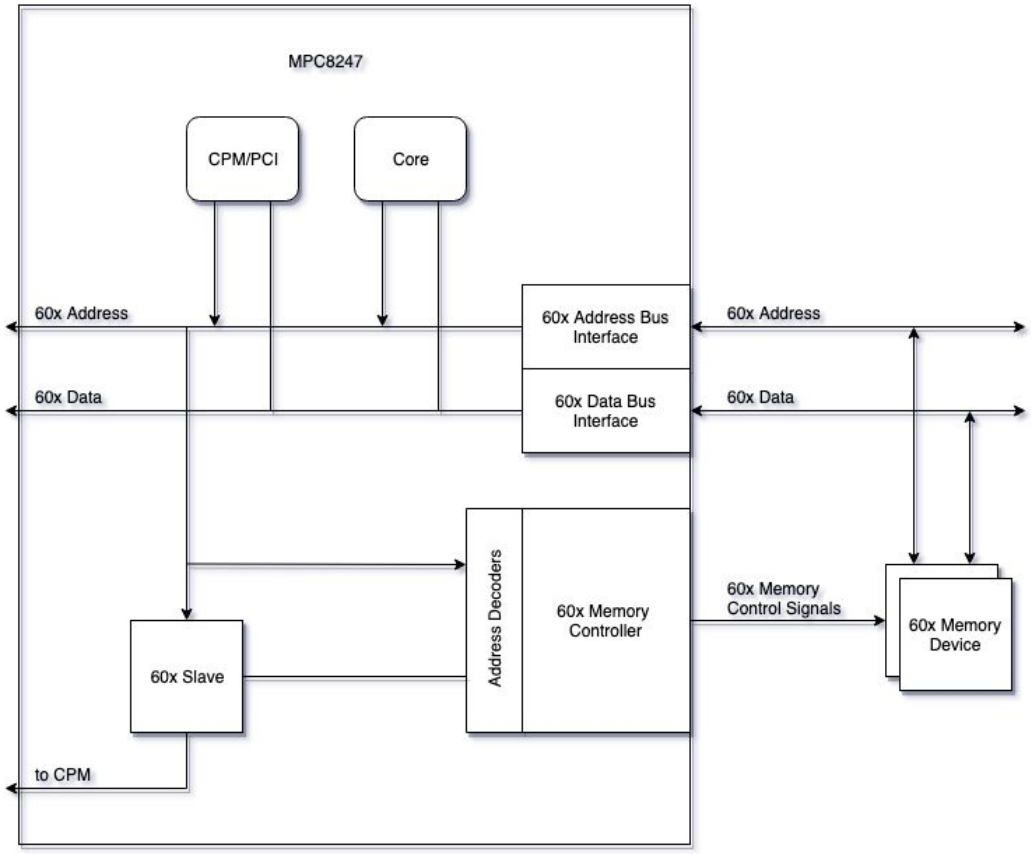


Figure 12. Simplified illustration of the Memory Controller in MPC8247. /5/

3 MEMORY FAULT MODELS

The purpose of memory design and test is to produce good memories and to deliver them to the customers. The goal of designing a memory is to result in a robust circuit and when the memory is tested the circuit is examined for manufacturing defects. /2/

The whole purpose of testing is to detect manufacturing defects. Characterization means examining the margins of a design and performing design verification. It looks at the marginality with respect to the voltage, timing, temperature and other environmental conditions. This is done to verify that the design is a good design. As a chip design is fabricated, it must be characterized to ensure for its robustness and that the design functions correctly. /2/

A fault model is a description of the way something can fail and there are numerous fault models that are applicable to memories. These memory fault models are described via the Markov diagram. Figure 13 shows a Markov diagram of a fault free memory cell. "R" describing a read operation, W_0 describing a write "0" operation and W_1 respectively a write "1" operation. S_0 and S_1 are used to indicate the cell state. /2/

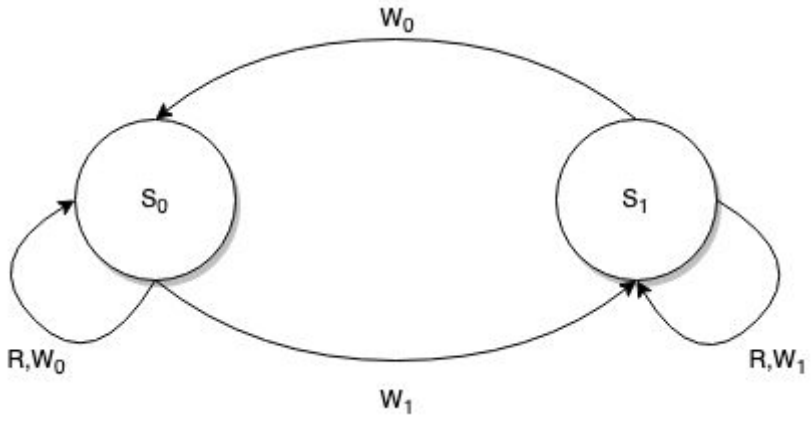


Figure 13. A fault free memory cell.

3.1 Single Cell Faults

The most common memory fault model, Stuck-at Fault (SAF), indicates that a cell is locked to in one state or another. Figure 14 shows a Markov diagram of a SAF in a memory. /2/

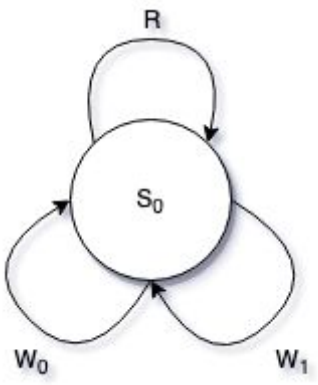


Figure 14. A Stuck-at Fault cell.

A Stuck Open Fault (SOF) means that the memory word cannot be accessed, because when the SA contains a latch during a read operation the previously read value may be returned. This fault can be modelled as SAF if the different amplifier behaves as a buffer.

A Transition Fault (TF) model looks like a Stuck-at Fault, except the memory cell will retain either state, “1” or “0”. After it is written to one state it cannot transition back to the previous state. Figure 15 shows a Markov diagram of the Transition Fault. /2/

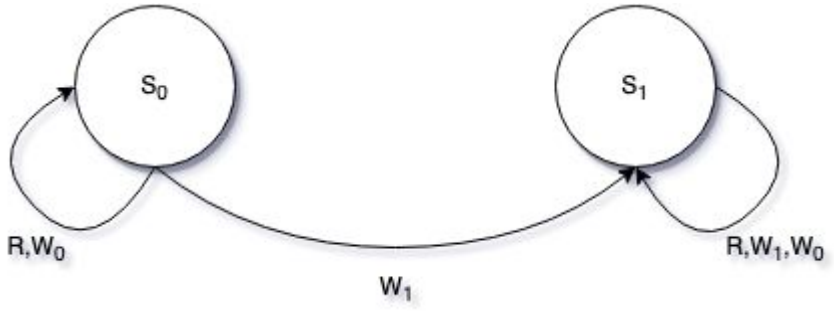


Figure 15. A Transition Fault.

A Read Destructive Fault (RDF) occurs when a read operation is performed to the cell causing an inversion of the value and returns an incorrect value. It can occur when cell is “1” or “0”. See Figure 16 for illustration when “0” is read from the memory cell flipping the memory cell value to “1”. /2/

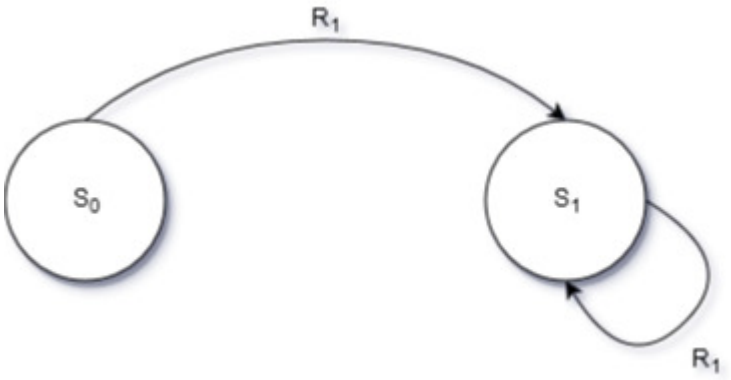


Figure 16. A Read Destructive Fault.

A Deceptive Read Destructive Fault (DRDF) occurs when a read operation is performed to the memory cell causing inversion of the cell value but returning the correct value. See Figure 17 for DRDF when a memory cell in state “0” is read, “0” turns in to “1” but returns “0”. /6/

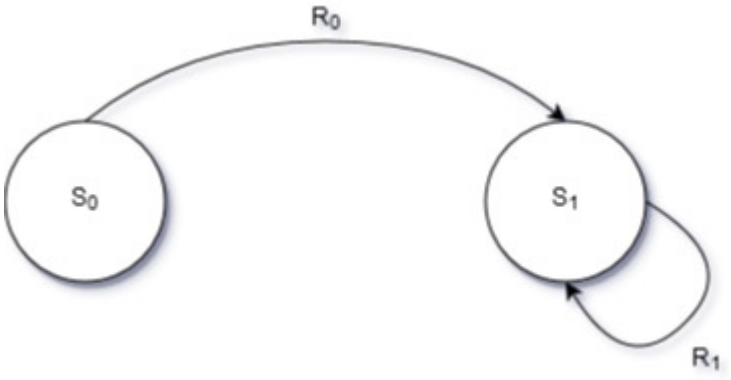


Figure 17. A Deceptive Read Destructive Fault.

A Write Destructive Fault (WDF) occurs when a write operation is performed to the cell causing the memory cell to flip. It can occur when the cell state is “1” or “0”. See Figure 18 for illustration when “0” is written on memory cells value “0”, resulting in the memory cell turning to “1”. /2/

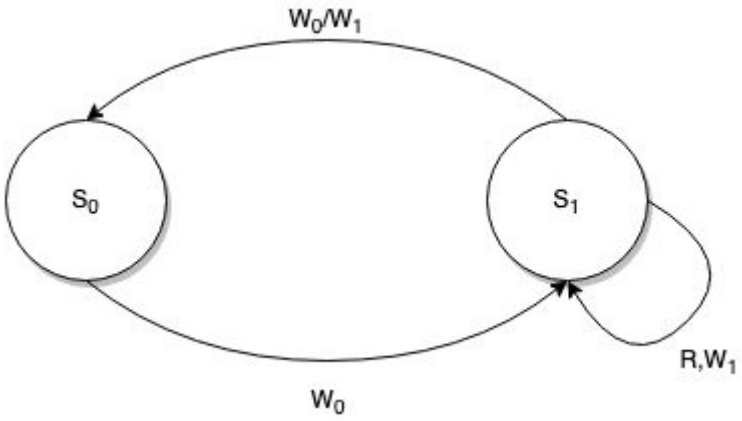


Figure 18. A Write Destructive Fault.

3.2 Multiple Cell Faults

There are multiple types of Coupling Faults (CF) models. To simply describe Coupling Fault (CF) models, a cell can couple into its neighbour cell thus causing it to falsely transition or to go to erroneous state. Two detect free cells are illustrated below in Figure 19. /2/

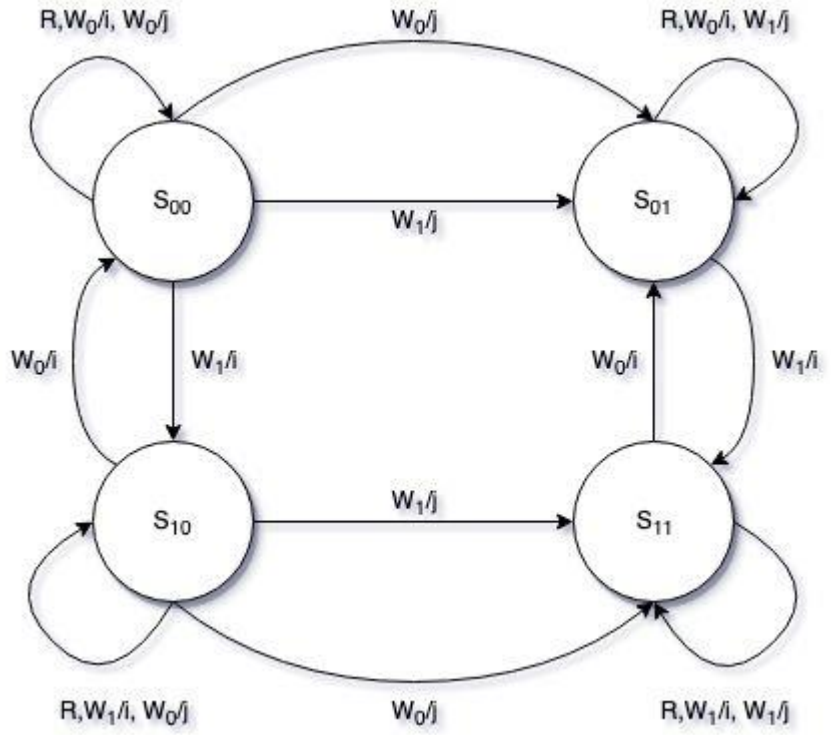


Figure 19. A pair of detect free cells.

Unidirectional coupling, where one call can couple into another cell without the opposite happening, is also possible. The cell doing the coupling is referred to as the aggressor cell and the cell that transitions is called the victim cell. The aggressor cell appears to operate correctly but the victim cells turns to incorrect state. This type of detect usually occurs when a parasitic diode, which is connected by a defect diode, is pulled high. On reverse operation the victim to aggress diode connection is reversed biased resulting in no transition in the aggressor cell. A bidirectional defect caused by a bridging defect can cause this reverse operation to occur. See Figure 20 for illustration. /2/

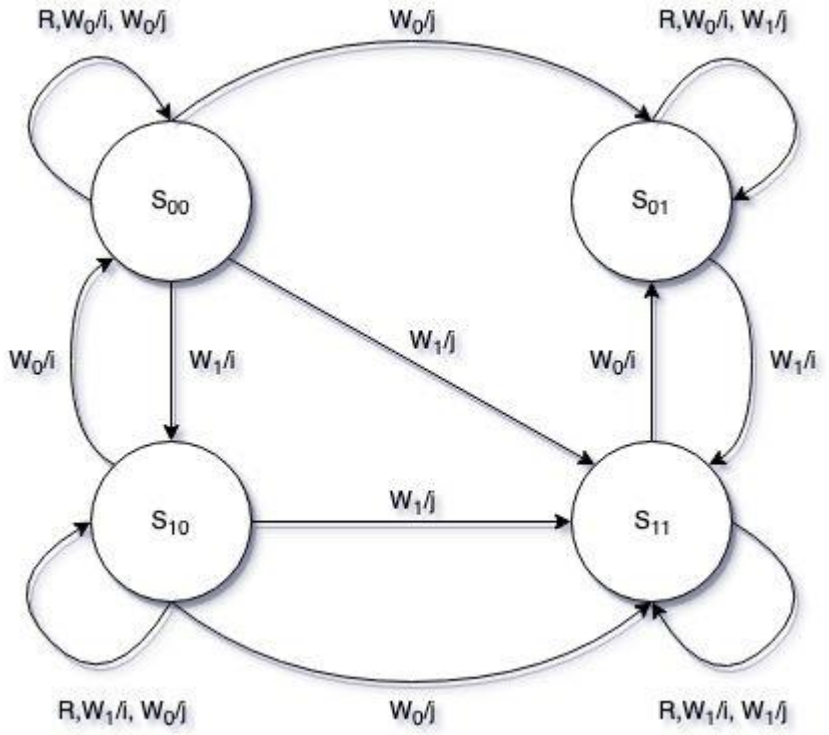


Figure 20. A coupling fault.

In the Neighbourhood Pattern Sensitive Fault (NPSF) model a memory cell is dependent upon the cells in its neighbourhood. Usually memories are described in terms of a nine-cell neighbourhood (see Figure 21). The base cell in the centre is surrounded by eight cells and it could be dependent on all or some of the neighbouring cells. The neighbourhood is the base cell plus those around it that can affect its behaviour. /2/

Neighbour Cell	Neighbour Cell	Neighbour Cell
Neighbour Cell	Base Cell	Neighbour Cell
Neighbour Cell	Neighbour Cell	Neighbour Cell

Figure 21. A nine-cell neighbourhood.

4 ELECTROMAGNETIC COMPATIBILITY

Electromagnetic Interference (EMI) is a serious and increasing form of environmental pollution. Its effects range from minor annoyances such as crackles on broadcast reception to potentially fatal accidents due to corruption of safety-critical control systems. EMI can cause electrical and electronic malfunctions, can prevent the use of the radio frequency spectrum, can ignite flammable substance atmosphere or can even have a direct effect on human tissue. /8/

4.1 System Defect

The use of microprocessors has stimulated the upsurge of interest in Electromagnetic Compatibility (EMC). Microprocessors are widely responsible for generating radio frequency interference and are susceptible to many interfering phenomena. Also, the widespread replacement of metal chassis by plastic enclosures has significantly reduced the protection offered to circuits by their housings. /8/

Processor-based control systems have taken over many functions which were earlier dominated by electromechanical or analogue equipment such as relay logic. Rather than being hard-wired to perform a particular task, microprocessors rely on a digital bus-linked architecture in which many signals are multiplexed onto a single hardware bus under software control. This structure exposes the systems to interference, because of the low level of energy needed to change state, but the effects of interference are impossible to predict. Even a random pulse could corrupt the operation depending on its timing with respect to the internal clock. Continuous interference usually has no effect on the programs state as long as it remains below the logic level threshold, but the systems operation will be disrupted as the interference crosses the threshold. /8/

4.2 Electromagnetic Phenomena

The electromagnetic phenomena which can be expected to interfere with control systems are supply voltage disruptions, dips, surges and fluctuations, transient overvoltage on a supply, signal and control lines, radio frequency fields, electrostatic

discharge (ESD) and low frequency magnetic of electric fields. ESD is mainly problem in electronic production, because once the components are assembled into a unit, they are protected from it. But it must be kept in mind that an ESD transient can corrupt the operation of a microprocessor just as a transient coupled into the supply signal or signal ports can, without damaging the components. /8/

It is not always clear whether the system malfunctions due to faulty software or due to EMI. This problem is highlighted when handling real time systems; the transient coincidences of external conditions with critical software execution state can cause operational failure which is almost or completely impossible to replicate. The symptoms, system crashes, incorrect operations or faulty data, can be identical to those induced by EMI. Sometimes the only way to distinguish faulty software from poor EMC may be by characterizing the environment in which the system is run. /8/

4.3 Memory Corruption

Memory fault is one of the most common hardware causes for machine crashes in today's datacenters. To design, evaluate and model systems that are resilient against memory errors require a good understanding of the underlying characteristics of errors in DRAM. As the number of DRAM in servers keeps growing and chip densities increase, DRAM errors pose an even bigger threat to the reliability of future generations of systems. /9/

Noise induced by transient can damage memories such as RAM or flash due to them interfering with clock circuits or flash write voltages. Corrupted memory can be manifested as flash checksum error or the system can lose a functionality because of corrupted code or data in flash or RAM. Flash corruption can be permanent, or it can require power cycle or reprogramming to recover the default state. RAM corruption might require a power cycle or a reset to resume the systems normal operation. /7/

Subsystem failure can be temporary or permanent. Permanent damage is easy to detect, whereas temporary damage such as latch-up or memory corruption can be

recovered to normal operation state by power cycling the equipment. When the subsystem is subjected to an EFT (Electrical Fast Transients), it can be partially damaged while still being fully functional, but when stressed with by a power supply, high temperature or abnormal operating conditions, the damaged component can then fail permanently. This effect is usually hard to identify and solve. /7/

4.1 Soft Errors

The energetic-proton component of galactic cosmic rays can be the cause of anomalous bit errors observed in memory chips in satellites. The dominant two radiation sources that cause soft errors, also known as single-event-upsets (SEU), in IC's are alpha particles and cosmic neutrons. In soft error, the data is corrupted the device itself is not damaged. The density of circuit on chips has reached the point where a subatomic particle can upset the information stored in a memory cell. Even multiple-bit SER is possible as the density of the circuits keeps increasing. /10/, /13/, /14/, /15/

The manufacturers of DRAM memories have been forced to incorporate three-dimensional structures into their cell designs to save area. At least three different approaches to cell technology for storing the bit charge have been adopted: a trench with internal charge (TIC), a trench with external charge (TEC) and the stacked capacitor (SC). A total of 26 different memory chip types were evaluated for cosmic ray soft fail rate in a study (Ziegler, J., IEEE, Nelson, M., Shell, J., Peterson, J., Gerderloos, Carl., Muhfeld, Hans. & Montrose, C). The chips represented the three different cell technologies and the final Soft Error Rate of the chips showed that there appears to be a strong correlation between the cell design and the SER sensitivity with TEC cells being the most sensitive, TIC cell beings about 1500 times less sensitive. and the SC design appearing to be intermediate sensitive to cosmic rays. /12/

The cosmic radiation includes gamma ray photons, protons and heavier atomic nuclei. The nuclear particles have billions of electron volts of kinetic energy and when they collide with the nuclei of nitrogen and oxygen atoms in the upper atmosphere,

they smash them into energetic charged fragments, which hit other nuclei and cascade in an avalanche of particles down through the atmosphere to surface of the Earth. At ground, there is a shower of secondary particles (including high-energy neutrons, protons, electrons, muons and gamma rays) and it is the nuclear component of this secondary particle flux (neutrons and secondary protons which cause the soft errors in the semiconductor memories). When a proton or a neutron with kinetic energy greater than the binding energy of a nucleus collides with a silicon nucleus in the semiconductor memory, the reaction produces highly ionizing recoils and low-Z charged particles (alpha particles and protons). When an ion travels through the memory cell, it will interact with the electric circuit by disturbing the electrons in the lattice. The ions kinetic energy is then converted into electron/hole pairs along its track as it comes to a stop in the semiconductor and the resulting charge moves according to the local electric fields and by diffusion. It can then generate a transient current pulse in the circuit, which if large enough, will disturb the circuit and the information stored in the memory cell will be corrupted. /10/, /19/, /20/

To protect a memory chip from cosmic radiation, it should be located underground, or it should be shielded with meter of concrete or equivalent to significantly reduce SER. The amount of soft errors increase at higher altitudes and vice versa. /16/, /20/

5 CHOOSING THE TEST APPROACH

Four different kinds of approaches were considered for the test implementation: running the tests through JTAG Technologies' CoreCommander software, from the application mode, the primary or the secondary bootloader. Each approach has its advantages and disadvantages and this chapter is about choosing the right approach developing the memory tests.

5.1 The CoreCommander

The CoreCommander approach seemed to be the best at first, because the software makes testing easy. It grants direct access to memory and I/O (input/output) devices through its JTAG (Joint Test Action Group) debug interface and Memory and peripherals can be manipulated without software programming. Theoretically this means that only the memory controller and its clock would need to be initialized thus simplifying the testing process. This approach would have been the most straightforward, but unfortunately the idea was shelved, because the PowerPC microprocessor used in the embedded device does not support CoreCommander.

5.2 The Application

The second idea was to run the tests in the application mode, because all peripherals such as clocks, controllers, file system (FS), communication and display interfaces are initialized. This idea was discarded, because it forces the developer to familiarize himself with millions of lines of the platforms source code, so the necessary modifications could be made without exposing the system to faults. This method also restricts the memory addresses CPU (Central Processing Unit) can write to. In addition, modifying memory controller options during runtime could lead to unexpected behaviour. It would also require comprehensive testing to ensure it is safe and reliable to use.

5.3 The Secondary Bootloader

The third approach is called secondary bootloader approach. It seemed to be ideal candidate for testing, because peripheral (such as LCD, communications) initialization takes place there. Test results could be written to LCD display and to the file system after they are initialized. This option would not force the developer to familiarize himself with the platforms source code thus saving time. This approach was abandoned as well because the secondary bootloader runs from the RAM.

5.4 The Primary Bootloader

The fourth and final option was the primary bootloader. It was avoided, because the test result reporting was deemed to be clumsy; the results must be read from the CPU registers with a debugger. But it was the only viable option, because primary bootloader runs from flash memory allowing unrestricted RAM access. Similarly, to bootloader 2 approach, this option allows the developer to ignore most of the platforms source code.

5.5 The Result

A combination of primary and secondary bootloader approaches was chosen, because there already are memory tests in the application mode and bootloaders tests need more diversity, such as different testing techniques and varying options in the memory controller. The test results are also written to the flash memory, so the secondary bootloader displays and stores them. This method also streamlines the use of this software, because there is no need to use a debugger to read the test results.

6 DEVELOPMENT ENVIRONMENT

Freescale CodeWarrior 8 was used to develop the software. It is an IDE (Integrated Development Environment) that allows editing the program code, building and debugging. The hardware debugger used was Lauterbach PowerDebug USB 3 with an extension cable (bottom right in the Figure 22). It is connected to 16-pin JTAG connector and the microprocessor is controlled with the TRACE32 software. TRACE32 can stop the execution of the program, step over, step in/out, and access memory.

Figure 22. The full setup.

The CPU module (see Figure 23) is attached to the embedded device. The microprocessor is located at the centre of the CPU module and RAM at the top right corner. The second RAM chip is on other side of the PCB (Printed Circuit Board). The other distinct IC's populated on the PCB are FPGA (Field-Programmable Gate Array) on the left and flash memory on the right, below the RAM.

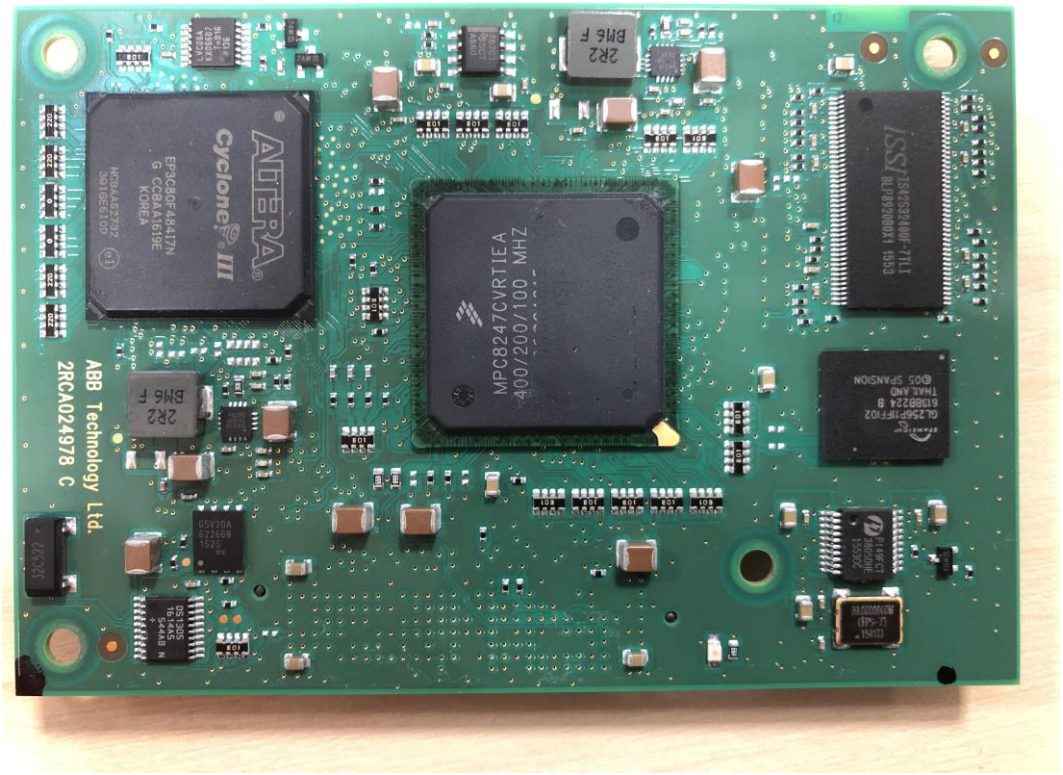


Figure 23. The CPU module.

7 FAULT RECORDING

Memory tests are run during the primary bootloader, which does not have access to LCD or FS. To record faults, a workaround was needed; log faults in to flash memory, so any further RAM tests would not corrupt the results.

7.1 Writing and Erasing the Flash

The microprocessor is configured to execute from external NOR flash ISSI IS29GL256. The device datasheet states that the flash memory must be in read mode, so it can be read /4/. Otherwise it would result in corrupt operational code (opcodes) when the CPU executes from it. As workaround, part of a subroutine responsible of writing the flash is executed from the Instruction Cache. The Instruction Cache loads 8 instructions (instruction size is 32 bits) from flash at once, so the code after the flash write is 8 word aligned by placing the function at static memory address with the use of linker script.

As explained in the Chapter 3, flash memory cannot be rewritten, it must be erased first and as mentioned above, the CPU cannot execute from flash while it's being written. The erase function is too big to execute from the cache, so the subroutine responsible of erasing the flash sector is copied into a bank in the CPU's inner dual-port RAM (DPRAM) and executed from there by manipulating the Instruction Pointer register. This dual-port RAM, which is used as a communication buffer, is different and a lot smaller than the external RAM. /5/

The software occasionally failed to boot to the secondary bootloader, presumably because writing the flash memory was still causing problems. Perhaps the absence of the delay needed after writing or erasing the flash memory that caused the crashes. It was then decided to escape the sunk-cost fallacy trap by developing an alternative approach. The existing project was studied more to discover that there was enough space for test results in DPRAM after all. After these changes the microprocessor booted successfully.

7.2 Timestamp

The tests are run for determined time. To achieve this, the microprocessor keeps track of time with its Time Base register pair TBU:TBL, referred as TB register. TB register indicates time since the power-up. The CPU bus clock is configured to 99 MHz frequency and the TB frequency can be calculated as follows:

$$TB\ frequency = \frac{Bus\ clock}{4} = \frac{99\ MHz}{4} = 24{,}75\ MHz \quad /5/$$

The next step is to calculate time to increment the TB register. The time base is calculated as follows:

$$Timebase = \frac{1}{TB\ frequency} = \frac{1}{24{,}75\ MHz} = 40{,}4\ ns \quad /5/$$

To calculate the elapsed time in seconds, the following equation is used:

$$Elapsed\ time = TB\ ticks\ \times Timebase$$

At compile time the software can be configured to run for 1 to 7 days. Even longer test durations, such as weeks or even months, are possible.

7.3 Displaying the Test Results

After the timeout, short list of the test results can be seen on the LCD. The comprehensive list of the test results could be seen in a Hypertext Markup Language (HTML) file in the FS. The HTML file describes the tests and lists the results in a HTML table element.

8 TEST PROCEDURE

In the test software, the device is designed to be stationed inside hot air oven to expose its memory IC's to possible faults. The test duration is lengthy, because the device must be tested in varying temperatures and it can take several weeks or even longer for a defect to occur. The source code for the memory tests is included in the Appendix 1.

8.1 Overview

The test software is written amongst the platforms source code and the memory tests are run at the bootloader section during the program startup. The tests begin by initializing the CPU and its peripherals. The memory controller needs to be configured (see Chapter 4) for correct RAM and flash memory accesses as well.

Memory tests are run for a programmed duration in loop and after each loop cycle the memory controller is reconfigured. If a fault is found during testing, it is logged by writing to free address space in the DPRAM.

After timeout occurs, the testing exits and the device boots to the secondary bootloader mode. The secondary bootloader reads the test results from the DPRAM and writes them into the devices FS and LCD. The test results include fault time (seconds from boot), test number, fault address, faulty, expected bits and test specific values. See Figure 25 for the flowchart of the program execution.

The CPU executes the program code from the flash memory, which has very slow access times (70 ns for IS29GL128) compared to RAM (~6 ns for IS42S32400F). The CPU is bottlenecked by the flash memory access times and it cannot use the RAM during the tests, so to speed up the program execution the software loads the instructions from a fast SRAM, the Instruction Cache. The CPU fetches two instructions from the Instruction Cache each clock cycle to reduce memory access times. /1/, /5/, /11/

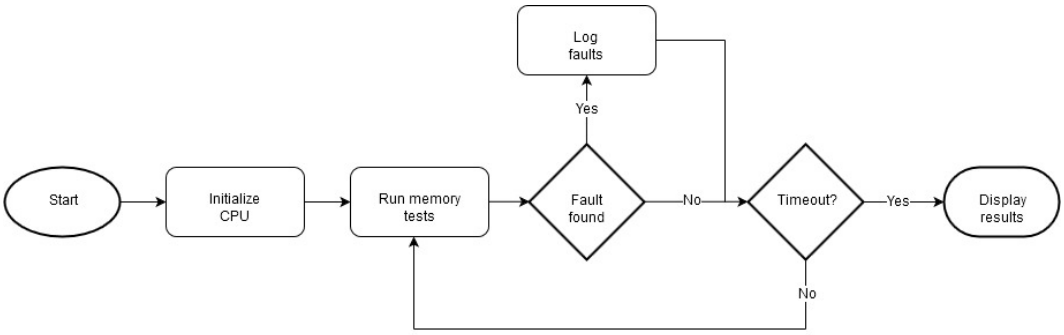


Figure 24. The memory test flowchart.

Optimization techniques such as instruction scheduling, and loop unrolling were applied on the hot loops of the memory tests to really stress the memory. Even though the microprocessor core has an excellent Branch Processing Unit (BPU), even the branching was optimized to ensure that branch shadowing is minimized. /5/

8.2 Hot Air Oven

The maximum recommend operating temperature of MPC8247 is 105°C. To calculate theoretical maximum temperature for the hot air oven, we need calculate ambient temperature T_A. $R_{\theta JA}$ is package junction-to-ambient thermal resistance (ºC/W) and P_D is thermal dissipation. Normal airflow is expected.

$$T_A = ambient\ temperature\ (°\mathrm{C})$$

$$T_J = 105°\mathrm{C}$$

$$R_{\boldsymbol{\theta} JA} = 21\ °\frac{\mathrm{C}}{\mathrm{W}}$$

$$P_D = 2{,}1W$$

$$T_A = T_J - \left(R_{\boldsymbol{\theta} JA} \times P_D\right) = 60{,}9°\mathrm{C}$$

The theoretical maximum operating temperature for the microprocessor is 60,9°C. As pointed earlier in the Chapter 6, the CPU module has other components in addition of the MPC8247. Two SDRAM memories, flash memory and other IC's

are included as well and add to the ambient temperature as the module is located inside a closed case. /18/

8.3 Test Descriptions

The first test tests for noise when a large number of bits change all at once by writing 1's, then 0's, then 1's again and reading the value and finally resetting the memory back to 0's. If the results differ from written values, then the memory is faulty. All memory the accesses are word size (32-bit). It also tests for Stuck at Fault.

The second test tests for Coupling Faults and Neighbourhood Pattern Sensitive Faults by writing to a memory address0 and then reading its neighboring cells (memory address). The result should always be 0, else the memory is defective. The neighbouring offsets of addresses to test are ±4 and ±1024. The memory is tested by writing words. Neighboring row memory addresses are calculated as follows:

$$RAM\ size = \frac{256\ Mb}{8} = 32\ MB$$

$$Rows = 8192$$

$$Banks = 8$$

$$Row\ size = \frac{\frac{RAM\ size}{Banks}}{Rows} = 1024\ B$$

The third test tests for Write Destructive Faults by performing transitioning write, then performing non-transitioning write and finally reading the result. If the value read is invalid, it means the memory cell value was flipped during a write operation. This test is performed with both writing 1's and 0's. All memory accesses are word size in this test.

The fourth test, tests for Read Destructive Faults by setting a value, then performing the first and then the second read. If the result is invalid, it means that the first read operation caused inversion of the cell value. Memory accesses are word size in this test.

The fifth test is a simple test. It aims to test that memory holds the values long enough. This test excels when modifying the memory controller values (refresh rate, for example). It is done by filling memory with values, then reading the values. If the read results do not match to written values, then the memory is deemed faulty. The test is performed using word size memory accesses.

The sixth test tests the memory by writing alternating bit patterns, first writing 0x555 type pattern, reading it, writing 0xAAA type pattern, reading again and then verifying the results.

The seventh test writes memory with large number of alternating bits, toggling the bits and writing again and finally toggling back and writing. The memory is faulty if the read result doesn't match the last write. Memory accesses are word size in this test.

The eight test writes the memory with unique byte size values and then verifies them. The write begins from value 0x40 and decrements by one after each write. When the test value reaches 0x00, it rolls over and continues from 0xFF.

The ninth test writes the memory with unique halfword (16-bit) size values and then verifies them. The write begins from value 0x8000 and decrements by one after each write. When the test value reaches 0x00, it rolls over and continues from 0xFFFF.

8.1 Reconfiguring the Memory Controller

The memory controller is configured to use specific values at boot time. By rotating different options in the SDRAM Mode Register, it is possible to expose the RAM to failure. The purpose of this is to modify wait state durations and clock cycle counts to find out whether the RAM chip exceeds the limits or not. Reconfigurable values are listed in Table 1.

Table 1. Reconfigurable values for SDRAM Mode Register (PSDMR). /5/

Name	**Description**	**Timing**

RFRC	Refresh recovery defines the earliest timing for an activate command after a refresh command.	5 clock cycles
PRETOACT	Precharge to activate interval defines the earliest timing for activate/refresh command after an activate command.	1...8 clock cycles
ACTTORW	Activate to read/write interval, which defines the earliest timing for read/write command after an activate command.	2...8 clock cycles
LDOTOPRE	Last data out to precharge defines the earliest timing for precharge command after the last data was written to SDRAM.	0...2 clock cycles
WRC	Write recovery time set the earliest timing for precharge command after the last data was written to the SDRAM.	1...4 clock cycles

All RAM tests are run by setting each register fields value to the minimum, and the maximum then resetting back to the default state one by one and by setting the register fields all at once. The memory tests are also run when the memory controller is in its default state. The default state in this context means the state, where the device is configured to operate when it is deployed.

The Memory Controller also possesses registers to manipulate the refresh timer period: 60x Bus-Assigned SDRAM Refresh Timer (PSRT) and Memory Refresh Timer Prescaler Register (MPTPR). Each PSDMR configuration is ran with three different refresh timer intervals: the default, minimum and maximum. The formula to calculate the refresh timer period is listed below.

$$TimerPeriod = \frac{(PSRT + 1) \times (MPTPR \rightarrow PTP + 1)}{BusFrequency} \quad /5/$$

The following equation is used to set the lowest refresh interval:

$$= \frac{2 \times 3}{99\ MHz} = 606\ ns$$

To set the highest refresh interval, the following equation is used:

$$= \frac{256 \times 256}{99\ MHz} = 662\ \mu s$$

9 ANALYSIS OF THE TEST RESULTS

The software was tested by running it for varying durations and with different CPU modules. A valid CPU module was tested multiple times to ensure that the software does not pick up false positives and all the tests were performed without hot air oven. The test results are included in the Appendix 2.

9.1 CPU Module 1

By looking at the tests results in Table 2, one can notice that the three least significant hexadecimal digits are always the same in the address column. It must also be noted that the result bits differ from the expected bits by 1 hexadecimal digit. In hexadecimal notation "F" equals to a nibble (a four-bit aggregation) "1111" when converted to binary and "B" equals to a nibble "1011" in binary. This information suggests that a bit line could be broken in one of the memory banks.

Two memory IC's are mapped to the microprocessor's memory and each memory circuit has 4 banks. The total memory in hexadecimal presentation is 0x02000000, so it has to be divided by 8 to get a bank size of 0x400000. As it can be seen from the test results, all memory faults occurred between 0x01600000 and 0x02000000, which effectively reveals that only a single memory bank is defective. /11/

$$Total\ memory = 1\ Mb \times 32\ \times 8 = 256\ Mb = 32\ MB = 0x02000000$$

$$Bank\ size = \frac{32\ MB}{8} = 4\ MB = 0x400000$$

The memory errors were only caught by RAM Test 5, which was described in Chapter 8.2. As the values are not held in the cell long enough, the problem is in the memory's refresh controller logic or the microprocessors Memory Controller. The refresh controller of the memory handles refreshing all the memory banks, so it is possible that a latch or a line between the memory and the refresh controller is broken causing the failures. Another odd phenomenon is that the memory faults are not guaranteed to occur, but when they occur multiple faults appear. /11/

The defects occurred when RFRC parameter was set to 5, which translates to sending activate command 5 clock cycles after the refresh command. It could mean that the refresh operation is in process when the activate command is sent and some cells could be uncharged.

It is hard to decipher if the problem is the bit line, the memory's refresh logic, the microprocessor or the combination of the previous possibilities. To further narrow down the possibilities, swapping out the memory or the microprocessor would rule out the possibility of defective microprocessor. The final step in this investigation would be to open the memory IC and inspect it with microscope to validate findings, but it is not necessary as the defective memory can now be sent back to the manufacturer for further investigation.

9.2 CPU Module 2

When the software was run on the CPU module 2, the defects always occurred in 10 minutes of the boot and the test result area in DPRAM was filled before timeout. The truncated test results are visible in Table 3.

Some of the fault addresses were different between boot ups, but the faulty bit was exactly the same and the three least significant hexadecimals were always the same on the fault address. The same analysis as in the previous chapter about the address and the bit line applies here as well.

The faults occurred when the PRETOACT or LDOTOPRE parameter was modified. No faults were found when the same test was run, but with modified Refresh Recovery parameter. The PRETOACT parameter modifies the timing for activate or refresh command after the precharge command and the LDOTOPRE parameter in turn defines the earliest timing for the precharge command after the last data was read from the memory. The parameters are linked to the same command, so it is natural that both expose faults of the defect memory chip.

The software found faults from the memory, but the memory addresses were always different. The fault addresses expanded across two memory banks, which could signal about malfunction with the refresh command. The fault could reside in either

the microprocessor or the memory chip and the next step would be to replace one of them.

9.3 CPU Module 3

CPU Module 3 was tested for 7 days and only two memory faults were detected. Both faults occurred at the same memory address and bit. The second fault occurred around 16 hours after the first one. To confirm the test results, the test should be rerun to see if defects still occur at the same memory location. The test results are visible in Table 4.

9.4 CPU Module 4

CPU Module 4 was tested for 7 days as well. 67 defects were found in total and truncated list of the test results are listed in Table 5.

Two least significant hexadecimal digits in the address were identical in all the test results and the same memory addresses repeat in the test results over the 7-day period. It should be noted that the functionality of the tests eight and nine is the same only difference being the memory access width, yet only the byte size memory access test caught the defects. The test should be rerun to confirm the test results.

The software provides strong evidence that the RAM is faulty as the fault occurs multiple times at the same addresses, yet even better, at the same bit.

10 CONCLUSIONS

The primary aim of this project was to implement testing software for ABB Oy Distribution Solutions Product Quality department, and this was clearly fulfilled as the software finds faults as designed. The useability of this project in customer product version is limited as updating the primary bootloader via remote means is too complicated for the device operators, so the software cannot reliably be distributed to deployed products.

However, there are further improvement ideas. The Product Quality department would benefit from solving the problem mentioned above as it would verify whether the memory is valid or not without sending it back to the factory. Secondly, the project could also be improved by implementing more tests as needed. In addition, the tests results could be logged in .csv format, so they could be opened in Microsoft Excel. Finally, the software is portable between the same family microprocessor cores, so in theory it could be expanded to other product versions.

REFERENCES

/1/ Jacob B., Ng, S. & Wang D 2008. Memory Systems: Cache, DRAM, Disk 1st Edition

/2/ Adams, R D. 2006. High Performance Memory Testing: "Design Principles, Fault Modeling and Self-Test"

/3/ Horowitz, P. & Hill, W. 2015. The Art of Electronics, Third Edition

/4/ IS29GL256. Integrated Silicon Solutions Inc. Accessed 18.11.2019. www.issi.com/WW/pdf/IS29GL256_128.pdf

/5/ MPC8272 PowerQUICC II™ Family Reference Manual. Freescale Semiconductor. Accessed 20.11.2019. https://www.nxp.com/docs/en/reference-manual/MPC8272RM.pdf

/6/ Hamdioui, S., Al-Ars, Z., VDG, Ad J. & Rodgers, M. 2004. Linked faults in random access memories: concept, fault models, test algorithms, and industrial results. Accessed 26.11.2019. https://ieeexplore.ieee.org/document/1291585

/7/ Design Considerations for Electrical Fast Transient (EFT) Immunity. Cypress Semiconductor. Accessed 20.12.2019. https://www.cypress.com/file/138636/download

/8/ Williams, T. 2001. EMC for Product Designers, Third Edition

/9/ Hwang, A., Stefanovici, I. & Schroeder, B. 2012. Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design

/10/ O'Gorman, T.J. 1994. The Effect of Cosmic Rays on the Soft Error Rate of a DRAM at Ground Level

/11/ IS42S32400F. Integrated Silicon Solutions Inc. Accessed 4.1.2020. http://www.issi.com/WW/pdf/42-45S32400F.pdf

/12/ Ziegler, J., IEEE, Nelson, M., Shell, J., Peterson, J., Gerderloos, Carl., Muhfeld, H. & Montrose, C. 1998. Cosmic Ray Soft Error Rates of 16-Mb DRAM Memory Chips

/13/ Hejmen, T. & Kruseman, B. 2005. Alpha-particle-induced SER of embedded SRAMs affected by variations in process parameters and by the use of process options

/14/ Satoh, S., Tosaka, Y. & Wender, S A. 2000. Geometric Effect of Multiple-Bit Soft Errors Induced by Cosmic Ray Neutrons on DRAM's

/15/ Baumann, RC. 2001. Soft errors in advanced semiconductor devices – Part I: the three radiation sources

/16/ O'Gorman, T.J., Ross, J., Taber, A., Ziegler, J., Muhlfeld, H., Montrose, C., Curtis, H. & Walsh, J. 1996. Field testing for cosmic ray soft errors in semiconductor memories

/17/ Pinheiro, E. 2009. DRAM Errors in the Wild: A Large-Scale Field Study.

/18/ MPC8272 PowerQUICC II Family Hardware Specifications. Freescale Semiconductor. Accessed 8.3.2020. https://www.nxp.com/docs/en/data-sheet/MPC8272EC.pdf

/19/ Normand, E. 1996. Single Event Upset at Ground Level

/20/ Ziegler, J. & Lanford, W. 1979. Effect of cosmic rays on computer memories

Table 2. CPU module 1 24-hour test results.

ID	Chip	Address	Bits	Expected	Notes	Timestamp
5	Upper	0x01CB7A60	0xFFFFFFFB	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Refresh Recovery (RFRC) 5 clock cycles.	0h 12m 19s from boot.
5	Upper	0x01DB7A60	0xFFFFFFFB	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Refresh Recovery (RFRC) 5 clock cycles.	1h 45m 38s from boot.
5	Upper	0x01CC7A60	0xFFFFFFFB	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Refresh Recovery (RFRC) 5 clock cycles.	1h 52m 49s from boot.
5	Upper	0x01D47A60	0xFFFFFFFB	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Refresh Recovery (RFRC) 5 clock cycles.	1h 52m 49s from boot.
5	Upper	0x01FF7A60	0xFFFFFFFB	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Refresh Recovery (RFRC) 5 clock cycles.	2h 21m 32s from boot.
5	Upper	0x01C01A60	0xFFFFFFFB	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Refresh Recovery (RFRC) 5 clock cycles.	2h 28m 42s from boot.

Table 3. CPU module 2 test results.

ID	Chip	Address	Bits	Expected	Notes	Timestamp
5	Lower	0x00FE99C4	0xFFFFFFBF	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Precharge To Activate interval (PRETOACT) 8 clock cycles.	0h 5m 11s from boot.
5	Lower	0x008099C4	0xFFFFFFBF	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Precharge To Activate interval (PRETOACT) 8 clock cycles.	0h 5m 12s from boot.
5	Lower	0x00FF99C4	0xFFFFFFBF	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Last Data Out To Precharge (LDOTOPRE) 0 clock cycles.	0h 5m 45s from boot.
5	Lower	0x00E919C4	0xFFFFFFBF	0xFFFFFFFF	Refresh Timer refresh period 606 ns (minimum). Last Data Out To Precharge (LDOTOPRE) 0 clock cycles.	0h 5m 45s from boot.

Table 4. CPU module 3 7-day test results.

ID	Chip	Address	Bits	Expected	Notes	Timestamp
2	Lower	0x006F0B70	0x00000001	0x00000000	Neighbour address: 0x006F0770 Neighbour bits: 0x00200000	56h 13m 18s from boot.
2	Lower	0x006F0B70	0x00000001	0x00000000	Neighbour address: 0x006F0770 Neighbour bits: 0x00200000	71h 52m 23s from boot.

Table 5. CPU module 4 7-day test results.

ID	Chip	Address	Bits	Expected	Notes	Timestamp
2	Lower	0x00AB3D14	0x00000001	0x00000000	Neighbour address 0x00AB3914 Neighbour bits 0x08000000	0h 3m 57s from boot.
2	Lower	0x008D9D14	0x00000001	0x00000000	Neighbour address 0x008D9914 Neighbour bits 0x08000000	0h 21m 31s from boot.
8	Lower	0x008FF914	0x0000005D	0x00000055	-	0h 25m 46s from boot.
2	Lower	0x00AE7D14	0x00000001	0x00000000	Neighbour address 0x00AE7914 Neighbour bits 0x08000000	3h 4m 51s from boot.
8	Lower	0x00B2F914	0x0000005D	0x00000055	-	14h 4m 14s from boot.
2	Lower	0x008D9D14	0x00000001	0x00000000	Neighbour address 0x008D9914 Neighbour bits 0x08000000	42h 8m 38s from boot.
8	Lower	0x008FF914	0x0000005D	0x00000055	-	61h 1m 30s from boot.
2	Lower	0x008D9D14	0x00000001	0x00000000	Neighbour address 0x008D9914 Neighbour bits 0x08000000	66h 16m 2s from boot.
8	Lower	0x008FF914	0x0000005D	0x00000055	-	90h 10m 27s from boot.
2	Lower	0x008D9D14	0x00000001	0x00000000	Neighbour address 0x008D9914 Neighbour bits 0x08000000	95h 24m 59s from boot.