OULUN SEUDUN
AMMATTIKORKEAKOULU

Dmitry Nikolaev

NAND-piirit standalone modem -alustoissa

NAND-piirit standalone modem -alustoissa

Dmitry Nikolaev
Opinnäytetyö
Syksy 2013
Tietotekniikan koulutusohjelma
Ohjelmistokehityksen
suuntautumisvaihtoehto
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Dmitry Nikolaev
Opinnäytetyön nimi: NAND-piirit standalone modem -alustoissa
Työn ohjaajat: Lari Manninen, Juha Alakärppä
Työn valmistumislukukausi ja -vuosi: Syksy 2013 Sivumäärä: 58

NAND-muistipiirien käyttö kasvaa vuodesta toiseen; muistipiirejä sovelletaan muun muassa kameroissa, muistitikuissa, mp3-soittimissa jne. Maailmassa on eri muistipiirien tuottajia ja useasti sovelluksen oikea toiminta riippuu muistipiirin valinnasta. NAND:n valintaan vaikuttavat monet eri aspektit, mutta tärkeimmät niistä ovat asiakkaan vaatimukset. Kuitenkin asiakkaalla voi syntyä kysymyksiä muiden valmistajien tuottamien muistipiirien käyttömahdollisuuksista. Tämä voidaan ennakoida ja suorittaa asiakkaan kysymyksiin vastaava tutkimus.

Valmistajat suorittavat eri testejä tuotannon prosessin aikana tehtaalla. Eri valmistajat käyttävät eri testausmenetelmiä ja tekniikoita, joten valmiin muistipiirin laatu voi erota valmistajasta toiseen. Renesas Mobile Corporationin toteuttamaan laitteeseen voidaan hyvinkin soveltaa kahden eri valmistajan muistipiirejä, eli Micronin ja Toshiban tuottamia piirejä. Siten työn päätavoitteena oli verrata kahden eri valmistajan piirejä keskenään ja haastaa tuottajien dokumentoimia muistipiirien parametreja. Tutkimuksen sivutavoitteena oli arvioida flash-operaatioiden ja Flash Translation Layerin toiminnallisuuden vaikutus muistipiirin elinaikaan.

Kaikki tutkimuksessa käytetyt testitapaukset on suunniteltu ja toteutettu johdonmukaisesti teorian avulla. Melkein kaikki tutkimuksessa käytetyt dokumentit ja ohjeet löytyvät internetistä; poikkeuksina ovat Renesasin ja Toshiban sisällä käytetyt paperit. Toteutuksessa käytettiin C-, C++- ja C#-ohjelmointikieliä. Lisäksi käytössä oli Renesasin toteuttamat työkalut: trace- ja debug-ohjelmisto sekä protolaitteita.

Testauksen tulosten näkökulmasta Micronin valmistama muistipiiri vaikuttaa paremmalta Toshiban tuottamaan muistipiiriin verrattuna. Toisaalta täytyy ottaa huomioon muistipiirin hintaa, kokoa ja datan käsittelynopeutta koskevat asiakkaan vaatimukset. Mainitun mukaisesti Toshiba alkoi vaikuttaa paremmalta: muistipiiri on 24 nm:n luokasta, piiri on suhteellisesti Micronin tuotetta halvempi ja datan käsittelynopeus kolminkertaistuu verrattuna Micronin. Tehdyn laskelman ja Program/Erase-testitapauksesta saadun datan perusteella voidaan todeta, että molempien valmistajien piirit kestävät hyvin aktiivista käyttöä. Voi myös sanoa, että ongelmat esiintyvät datan käsittelyssä todennäköisemmin asiakkaan valitseman muistipiirin elinajan loppuvaiheessa kuin Micronin tekemän tuotteen samassa elinajan vaiheessa.

Asiasanat:
NAND, SLC, miTron, muistipiirit, testaus, FTL, flash-muistipiirit

ABSTRACT

Oulu University of Applied Sciences
Information Technology and Telecommunications, Software Development

---

Author: Dmitry Nikolaev
Title of thesis: NAND memory chips in standalone modem platforms
Supervisors: Lari Manninen, Juha Alakärppä
Term and year when the thesis was submitted: Autumn 2013 Number of pages: 58

The use of the NAND memory has been growing up with every passing year; this type of memory chips is being used in almost every electronic device: digital cameras, flash-memories, mp3-players etc. There are many NAND memory chips manufactures. Under some circumstances the application correct behavior and functionality is directly dependent upon features of the NAND memory chip provided by the manufacturer. Nonetheless, the main aspect to be considered by the application's producer is the client's requirements. The client's questions and concerns about mentioned features should be foreseen. For this purpose some research should be done on the producer's side.

Different test cases are done within the NAND memory chips manufacture process. Every manufacturer uses its own test methods and techniques, thus, the ready to ship product's quality differs from manufacturer to another. For the application designed by the Renesas Mobile Corporation can be applied two different memory chips manufactured by the Toshiba and Micron. Thus, the research's main objective is to compare two different manufacturers' memory chips and challenge parameters documented by factories. The paper in question secondary objective is flash chip's operations and flash translation layer produced impact estimating on the memory chip lifespan.

Every single test case was designed and implemented consistently based on the theory. Almost all documents and manuals used in this paper can be found on the Internet – the only exception is Renesas Mobile Corporation and Toshiba internal documentation. For test cases implementation were used C, C++ and C# programming languages. During the tests execution trace and debug software and hardware provided by the Renesas were in use.

In respect of the tests results, the Micron memory chip looks better than the product produced by the Toshiba – the amount of the erroneous bits is about 20 times smaller, there is no current leakage etc. Consequently, the read or write operation's fault will happen most probably on the Toshiba's product. Nevertheless, as was mentioned above, the only aspect, which can be considered under such circumstances, is the client's requirements. Hence, the Toshiba memory chip is starting look better – it's smaller, relatively cheaper and three times faster (it has error correction code algorithm built-in) comparing to the Micron memory chip.

Keywords:
NAND, flash memory, miTron, FTL, SLC, testing

SISÄLLYS

1 JOHDANTO

Sulautetuissa järjestelmissä entistä enemmän käytetään NAND-muistipiirejä. Melkein jokaisessa nykyaikaisessa laitteessa käytetään mainittuja muistipiirejä: SSD-kovalevyt (Solid State Drive), muistitikut, kameroissa olevissa muistikorteissa jne. Muistipiirien hinta, koko ja datan käsittelynopeus vaikuttavat muistipiirin valintaan. NAND-muistipiiri on nopeampi, pienempi ja halvempi vastaavaa NOR-piiriä. Täytyy myös sanoa, että paitsi etuja NAND:lla on myös haittoja. Esimerkiksi datan satunnainen saatavuus on erittäin hidas muistipiirin arkkitehtuurin takia, sillä piirissä kaikki solut on kytketty sarjaan. Siksi NAND-muistipiirejä käytetään pääasiassa datan varastoksi. Jos muistipiirissä säilytetään laitteen firmware, käynnistyksen aikana ladataan koodien pätkät RAM-muistiin ja siten tuetaan datan nopeaa satunnaista saatavuutta.

Kuten jokaisessa piirissä, NAND:ssäkin on toimintaan liittyviä tiettyjä rajoituksia ja ongelmia, jotka johtuvat piirin arkkitehtuurista ja perusoperaatioiden suorittamisessa käytetyistä tekniikoista, kuten datan vuoto, kirjoitus- ja poistosyklien lukumäärä, luku-, kirjoitus- ja poistohäiriöt jne. Monet ongelmat voivat aiheuttaa virheitä lukuoperaation aikana. Muistipiirien valmistajat ovat kehittäneet eri menetelmiä ja toimenpiteitä ongelmien ratkaisemiseksi ja siten kehittäneet muistipiirin käyttämiseen liittyviä tiettyjä rajoituksia. Jotkin rajoitukset voidaan väistää hyvin suunnitellun ohjelmiston avulla. Esimerkiksi hyvä FTL-algoritmi (Flash Translation Layer) voi pidentää muistipiirin elinaikaa.

Valmistajat suorittavat eri testejä tuotannon prosessin aikana tehtaalla. Eri valmistajat käyttävät eri testausmenetelmiä ja tekniikoita, joten valmiin muistipiirin laatu voi erota valmistajasta toiseen. Renesasin suunnittelemassa laitteessa voi hyvinkin käyttää kahden eri valmistajan muistipiirejä, eli Micronin ja Toshiban tuottamia piirejä. Täten työn päätavoitteena oli verrata kahden eri valmistajan piirejä keskenään ja haastaa valmistajien dokumentoimia muistipiirien parametreja. Testattavana oli muun muassa muistisivun osittainen kirjoittaminen. Tavoitteena oli tarkastaa, kuinka monta kertaa pystytään kirjoittamaan samaa sivua osittain ennen virheiden esiintymistä. Tämä on todella tärkeä muistipiirin parametri tiedostojärjestelmän näkökulmasta. Tutkimuksen sivutavoitteena oli arvioida flash-operaatioiden ja FTL:n toiminnallisuuden vaikutus muistipiirin elinaikaan.

Kaikki tutkimuksessa käytetyt testitapaukset on suunniteltu ja toteutettu johdonmukaisesti teorian avulla. Melkein kaikki tutkimuksessa käytetyt dokumentit ja ohjeet löytyvät internetistä. Poikkeuksena ovat Renesasin ja Toshiban sisällä käytetyt paperit. Toteutuksessa käytettiin C-,

C++- ja C#-ohjelmointikieliä. Lisäksi käytössä oli Renesasin toteuttamat työkalut: trace- ja debug-ohjelmisto sekä protolaitteita.

2 MINI HOST DOMAIN JA μiTRON

Tässä luvussa kuvataan lyhyesti Mini Host domain ja μiTRON RTOS (Real Time Operating System).

2.1 Mini Host domain

Mini Host domain on kaupallisen järjestelmän osa. Siksi jätetään joitakin alueita ja osia pois järjestelmän kokonaiskuvauksesta ja muita kuvataan vain muutamalla sanalla. Mini Hostin päätarkoituksena on host-palvelujen tarjoaminen modem-domainiin (2G/3G/LTE) sekä viestinnän ohjaaminen PC:n ja APE:n (Application Engine) kanssa. Lisäksi Mini Host ohjaa HW-osia (Hardware), jotka eivät liity soludomainiin. Kuvattava domain käyttää etuoikeutettua (pre-emptive) monitaskien pohjaista reaaliaikaista käyttöjärjestelmää (RTOS), joka on suunniteltu μiTRON:n dokumentaation mukaisesti. (1, s. 11.)

Mini Host domain on esitetty kuvassa 1 ja siihen liittyvät seuraavat osat:

- Virranhallinta (Power Manager)
- Laitteen ajurit (Device Driver)
- USB (Universal Serial Bus)
- Tietoverkon protokollapino (Networking Protocol Stack)
- Viestintä (Intertask Communication)
- Firmwarehallinta (Firmware Management)
- Testaus (Test)
- Ydinrajapinta (Core interface)
- Secure
- Muisti ja muistinhallinta (Storage) (1, s. 11).

Virranhallinta-alijärjestelmä koostuu eri ajureista ja taskeista ja sen tehtävänä on virran ja kellojen ohjaus sekä idle-tilan toimintojen suorittaminen (2, s. 27).

Laitteen ajurit -alijärjestelmä on vastuussa Mini Hostin käynnistämisestä ja erilaisista I/O (input - output) -väylistä ja rekistereistä sekä keskeytysten ja poikkeusten (exception) käsittelystä (3, s. 13).

USB-alijärjestelmä vastaa viestien ja komentojen vastaanottamisesta ja käsittelystä PC:ltä sekä tietojen lähettämisestä PC:lle (1, s. 55).

Tietoverkon protokollapino -alijärjestelmä hoitaa pakettien vastaanottamista verkosta ja lähettämistä verkkoon. Lisäksi kyseessä oleva alijärjestelmä käsittelee viestintää PC:llä olevan HTTP (Hypertext Transfer Protocol) UI (User Interface) -ohjelmiston parissa. (4, s. 40.)

Viestintä-alijärjestelmä hallitsee signaalien käsittelemistä eri Hardware- ja Software-alijärjestelmien välillä (3, s. 14).

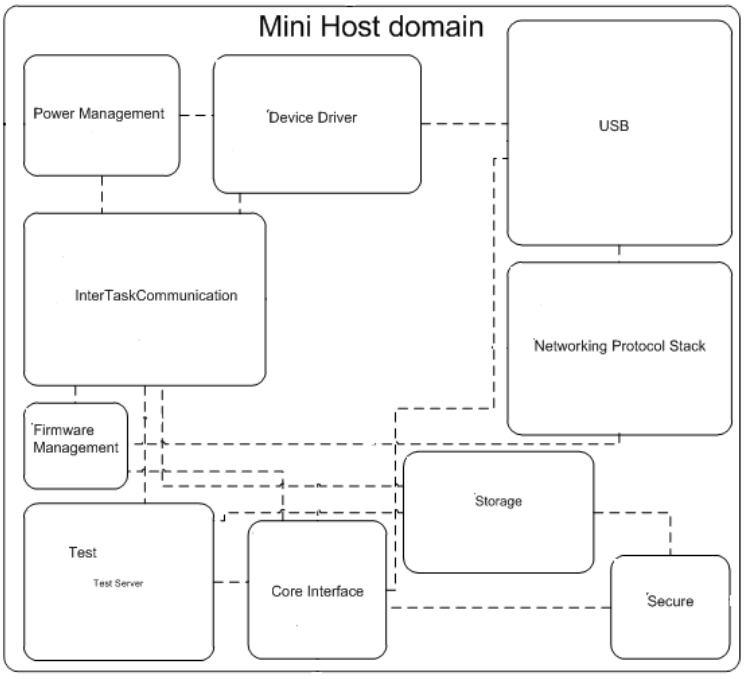
Firmwarehallinta-alijärjestelmä käsittelee PC:n pyyntöjä koskevia version ja tuotteen informaatiota. Lisäksi alijärjestelmä hoitaa järjestelmien käynnistämistä uudestaan sekä modeemin tilojen asentamista ja tarvittaessa tilojen informaatiota lähettämistä käyttäjälle. (1, s. 49.)

Testaus-alijärjestelmä käsittelee testaukseen liittyviä komentoja (1, s. 41).

Ydinrajapinta-alijärjestelmä tarjoaa rajapintaa IPC:lle (Inter-Process Communication) (3, s. 56).

Secure-alijärjestelmä käsittelee turvallisuuteen liittyviä ominaisuuksia, esimerkiksi autentifikaatiota ja laitteen erikoissertifikaatin luontia (1, s. 46).

Muisti ja muistihallinta -alijärjestelmä (Storage Subsystem) hallitsee datan kirjoittamista muistiin ja poistamista muistista (4, s. 21).



KUVA 1. Mini Host Domain (1, s. 70)

2.2 μiTRON

Kuten aikaisemmin on mainittu, Mini Host käyttää reaaliaikaista käyttöjärjestelmää, joka on suunniteltu ja toteutettu μiTRON:n spesifikaation mukaisesti. Mainitaan muutama fakta kyseessä olevasta spesifikaatiosta ja sen historiasta.

Tohtori Sakamura aloitti TRON-projektin (The Real-Time Operating System Nucleus) vuonna 1984 Tokion yliopistossa yrittäen luoda ideaalisen tietokoneen arkkitehtuurin. Projektin visiona on ollut fakta, että tulevaisuudessa jokainen laite on sulautettu järjestelmä, sekä laitteisto on yhdistetty keskenään isoon verkkoon helpottaen ihmisten arkielämää (Highly Functional Distribution System - HDFS). Täten TRON-projektin päätavoitteena on HDFS-järjestelmän toteutus. Projekti on jaettu muutamaksi aliprojektiksi, kuten ITRON (Industrial TRON, spesifikaatiot on tarkoitettu sulautetulle laitteistolle), BTRON (Business TRON, spesifikaatiot on osoitettu henkilökohtaisille tietokoneille ja työasemille), CTRON (Communication and Central TRON, käyttöjärjestelmän rajapinnan spesifikaatio, jonka päätavoitteena on viestinnän ohjaus ja informaation käsittely) ja TRON HMI (yleisspesifikaatio ihminen–kone-rajapinnalle). (5, s. 1–21.)

TRON:n toimintatapana on avoin spesifikaatio, eli kuka tahansa voi käyttää tutkimusten tuloksia ja rakentaa niiden perusteella omia projekteja ja järjestelmiä. Vuonna 1996 TRON on järjestänyt kyselyn, jonka tulosten mukaisesti päätettiin, että vain pieni osa insinööreistä voi käsitellä kaikkia käytössä olevia RTOS:iä. OS:n (Operating System) vaihtaminen vaatii todella paljon työtä ja aikaa insinööreiltä ja syynä on spesifikaation koko. TRON-projektissa oli päätetty toteuttaa RTOS:n yleisspesifikaatiota, jota voitaisiin helposti soveltaa monissa olemassa olevissa sulautetuissa järjestelmissä säästäen uuden käyttöjärjestelmän opetusaikaa käyttämällä yleiskonsepteja ja teknisiä termejä. (5, s. 1–21.)

Vaikein probleema yleisspesifikaation toteutuksessa on tasopainon löytäminen HW:n (Hardware) tarjoaman korkeimman suorituskyvyn ja SW:n (Software) kehityksen parantamisen väliltä. Pienten ja suurten sulautettujen järjestelmien vaatimukset eroavat toisistansa kovasti. Pienten järjestelmien suorituskyky kärsii, kun SW:n koko kasvaa sellaisten ominaisuuksien takia, joita ei tarvita. Toisaalta tarvitaan useasti aikaisemmin mainitut ominaisuudet suurissa järjestelmissä. Jokaiselle järjestelmälle voidaan määrittää sitä koskevia ominaisuuksia ja suorituskykyvaatimuksia. Toisaalta skaalautuvan spesifikaation luominen, joka sopisi eri sulautettujen järjestelmien vaatimuksiin, olisi hyvin käytännöllistä. RTOS:n spesifikaation on siis vastattava seuraavia vaatimuksia:

- Järjestelmän on saatava maksimaalinen suorituskyky HW:stä.
- Järjestelmän on oltava käyttökelpoinen SW:n kasvavassa tuottavuudessa (productivity).
- Järjestelmän on oltava skaalautuva.
- Järjestelmällä on avoin spesifikaatio, eli kuka tahansa voi toteuttaa omia järjestelmiä ja myydä niitä. (5, s. 1–21.)

Ensimmäinen ITRON:n spesifikaatio oli kehitetty vuonna 1987, ITRON1. Monet RTOS:t oli kehitetty tämän spesifikaation mukaisesti. Myöhemmin (vuonna 1989) ITRON-projekti on julkaissut kaksi spesifikaatiota: μiTRON2.0:n ja ITRON2:n spesifikaatiot. μiTRON:n spesifikaatio oli osoitettu pienille järjestelmille, joissa on joko 8-bittiset tai 16-bittiset MCU:t (Microcontroller Unit). Sen erityispiirteenä oli kernelin rajoitettu toiminnallisuus. μiTRON:n spesifikaatio oli toteutettu monissa laitteissa, joissa oli rajoitetut HW-resurssit, kuten muisti ja MCU:n suorituskyky. μiTRON:n spesifikaation oli oltava yllä mainittujen RTOS:n vaatimusten mukaisesti skaalautuva. Edellä mainitusta vaatimuksesta johtuen spesifikaatio oli laajennettu myös 32-bittisille MCU:lle. Vuonna 1993 oli julkaistu μiTRON3.0:n päivitetty spesifikaatio. Vuonna 1996 oli aloitettu toinen vaihe ITRON:n spesifikaation kehittämisessä ja sen tuloksena oli μiTRON4.0:n spesifikaatio. Tämä spesifikaatio pidetään 4. sukupolven spesifikaationa. On neljä syytä, joiden vuoksi μiTRON4.0:n spesifikaatio oli julkaistu:

- SW:n siirrettävyyden parantaminen
- SW-komponenttien toiminnallisuuden lisääminen
- uudet vaatimukset ja tutkimusten tulosten liittäminen
- HW:n kehityksestä johtuvien ominaisuuksien ja toiminnallisuuksien liittäminen (5, s. 1–21.)

Aikaisemmin oli mainittu, että verrattuna μiTRON3.0:aan spesifikaatioon uudet ominaisuudet lisättiin μiTRON4.0:aan. Uusiin ominaisuuksiin liittyvät muun muassa mutexit ja interrupt service -rutiinit. Lisäksi μiTRON4.0:n spesifikaatio vähentää toiminnallisuuden riippumista toteutuksesta. (5, s. 1–21.)

μiTRON4.0:sta voi kertoa paljon enemmän mielenkiintoisia faktoja, mutta tämä tutkimus ei ole μiTRON4.0:n spesifikaatiosta, vaan NAND-piirien testauksesta ja niiden toiminnallisuuden vertailua. Seuraavissa luvuissa kerrotaan tarkemmin NAND-piirien ominaisuudesta ja arkipäivien ongelmista, jotka liittyvät piirien käyttöön. Lisäksi pyritään kuvaamaan ongelmien ratkaisuja sekä ratkaisujen hyvät ja huonot puolet.

Testaus ja vertailu on tehty Renesas Mobile Corporationin ehdottamalla HW:llä ja aikaisemmin kuvatulla SW:llä käyttämällä Testaus- ja Storage-alijärjestelmiä. Kuitenkin täytyy mainita, että testitapaukset ja lisäohjelmisto piti toteuttaa tutkimuksen aikana. Salassapitosopimuksen mukaisesti ei saa julkaista ohjelmiston ja testitapausten lähdekoodia, eikä liittää laitteiston kuvioita ja laitteistoon liittyviä skeemoja. Projektin aikana käytettiin seuraavia ohjelmointikieliä: C, C++, C#. Tämän lisäksi trace- ja debug-ohjelmisto sekä debug-laitteisto oli käytössä.

3 NAND-MUISTI

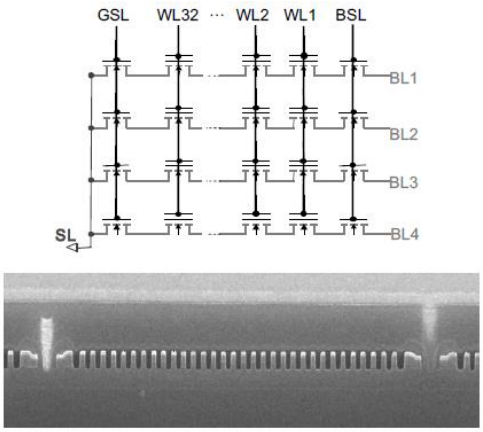
Tässä luvussa kuvataan NAND-flash-muistia sekä muistin arkkitehtuurista johtuvat eri muistin lajit (single ja multilevel cells, SLC ja MLC). Kuvataan muistipiirien eri ominaisuuksia ja niihin liittyviä ongelmia. Kerrotaan, mikä ero on NOR- ja NAND-muistipiirien välillä jne.

3.1 NAND-muistipiirin yleiskuvaus ja historia

NAND-flash-muisti on haihtumaton (non-volatile) muisti, joka ei tarvitse virtaa tietojen säilyttämiselle (6, s. 1).

NAND:n kehityksen päätavoitteina olivat yhteen bittiin liittyvien kulujen vähentäminen ja muistitilavuuden suurentaminen, jotta voitiin kilpailla HDD:n (Hard Disk Drive) kanssa. Muistipiiri ei vaadi paljon energiaa sen toimintaan verrattuna HDD:iin, sen koko on pienempi HDD:a ja NAND-tekniikka poistaa kokonaan mahdollisuuden tietojen häviämisestä laitteen rajun käsittelyn jälkeen. (8, s. 1.)

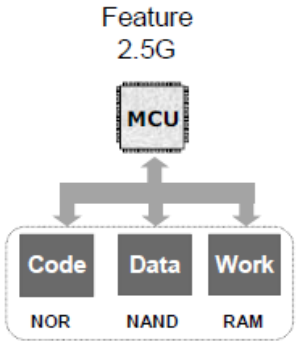
NAND-flash-muistipiirien historia alkaa vuonna 1984 Toshiban laboratoriossa. NAND:n idean keksijä on tohtori Fujio Masuoka. Toshiba on esittänyt NAND muistipiiri vuonna 1988. Kuvassa 2 on esitetty NAND-muistipiirin arkkitehtuuri. (7, s. 3.)



KUVA 2. NAND-muistipiirin arkkitehtuuri (7, s. 3)

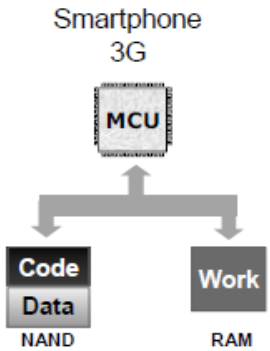
NAND:n kehitykseen vaikuttivat erittäin tehokkaasti digitaalisen valokuvauksen tekniikoiden kehittäminen sekä digitaalisten muistivarastojen käyttöönoton kasvaminen. Western Digital havainnollisti ensimmäisen 2,5”:n NAND SSD:n (Solid State Drive) vuonna 1989. Vuonna 1995 esiteltiin ensimmäinen irrotettava (removable) media, joka sisälsi NAND-muistipiirin (SmartMedia-kortti). Tämän laitteen ilmestymisen jälkeen oli julkaistu MultiMedia Card vuonna 1997, Memory

Stick vuonna 1998, Secure Digital vuonna 1999 ja x-D Picture Card vuonna 2002. Laitteiden pienoisversiot ”mini” ja ”micro” tulivat julkisuuteen jälkeenpäin. Ensimmäinen NAND SSD, joka otettiin käyttöön PC:lla (Personal Computer), esitettiin vuonna 2006. NAND-muistipiirien käyttäminen puhelimissa on alkanut 2,5G-sukupolvesta. Feature-puhelimissa käytettiin kolmea eri muistipiiriä rinnakkain: NOR-muistipiirejä koodin ajamista varten, NAND muistivarastona ja RAM työmuistina. Tämä arkkitehtuuri on laajennettu XIP-arkkitehtuuri (Execute in Place), joka sisältää datan säilyttämiselle tarkoitetun NAND-muistipiirin. Kuvassa 3 on esitetty Feature-puhelimen arkkitehtuuri. (7, s. 10–14.)



KUVA 3. Muistijärjestelmän arkkitehtuuri – Feature 2.5G (7, s. 10–14)

Älypuhelimissa koodin säilyttämisen funktio siirtyy NAND-muistipiirille ja sen ajaminen tapahtuu DRAM-muistissa (Dynamic Random Access Memory). Kuvassa 4 on esitetty älypuhelimen muistiarkkitehtuuri. (7, s. 10–14.)

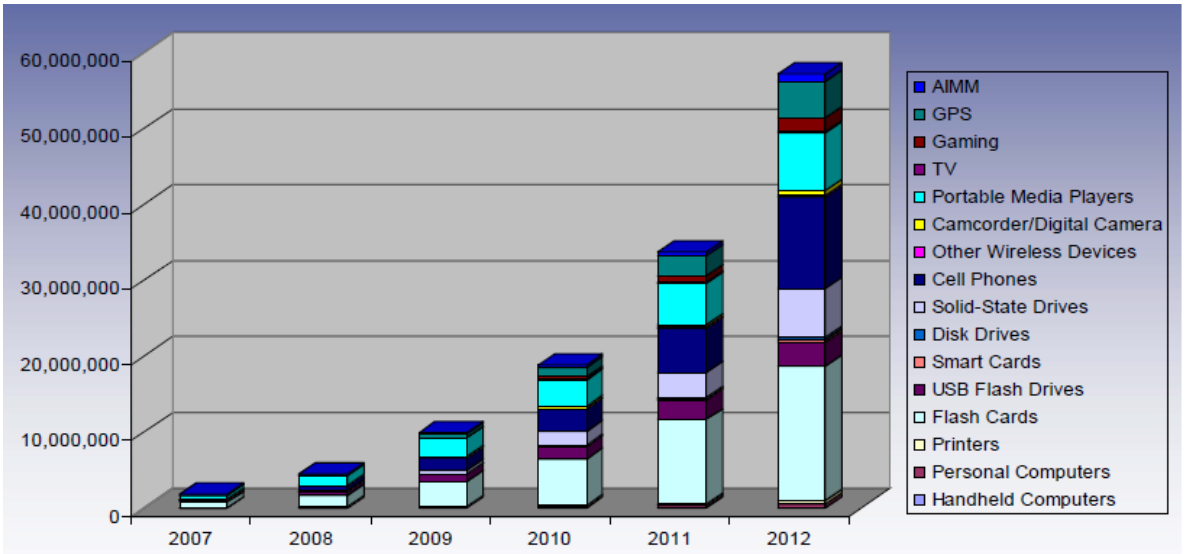


KUVA 4. Älypuhelimen muistiarkkitehtuuri (7, s. 10–14)

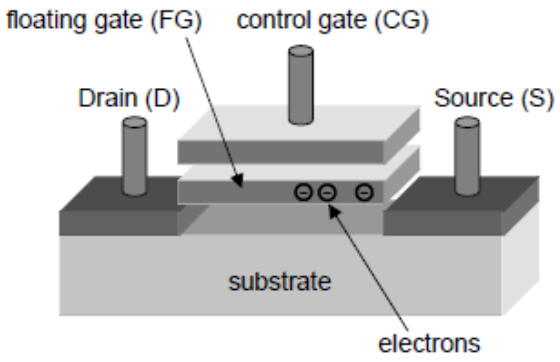
Kuvassa 4 oleva arkkitehtuuri on vastaavaa XIP-arkkitehtuuria halvempi, mutta samalla semmoisen arkkitehtuurin haitat ovat itsestään selvät. Energian kulutus kasvaa ja käynnistäminen

vaatii enemmän aikaa (ennen ajamista koodi ladataan DRAM:iin ja vasta tämän proseduurin jälkeen ajetaan ladatun koodin pätkää). (7, s. 10–14.)

Kuten voi ymmärtää aikaisemmin mainituista tiedoista, NAND-muistipiirejä käytetään laajasti elektroniikassa. Kuvassa 5 nähdään, miten muistipiirien käyttöönotto on kasvanut vuodesta toiseen. NAND-muistipiirit ovat halpoja verrattuna NOR:ään. NAND-piirin solun koko (kuvassa 6 on kuvattu Floating Gate -pohjainen solu) on pienempi NOR-piirin solua ($4F^2$ verrattuna $10F^2$) sekä jotkin operaatiot suoritetaan nopeammin, esimerkiksi tietojen poistaminen. Muistipiirin kokoon vaikuttaa piirin sisäinen arkkitehtuuri (arkkitehtuurista kerrotaan tarkemmin luvussa 3.3). NAND-piirien solut eivät vaadi erillisiä metallikontakteja. NAND-muistipiirit ovat samanlaisia kuin HDD datan osoittamisen ja säilyttämisen näkökulmasta. Kuten HDD:ssä, pyritään kirjoittamaan tiedostot peräkkäiseksi tavusarjaksi, esimerkiksi kuvia, videoita tai firmwaren koodia. Lisäksi voi toteuttaa satunnaisen datan saatavuuden lataamalla tietojen tavuja RAM-muistiin. Myös NAND-piirit vaativat ECC:tä (Error Correction Code) virheiden korjaamiseksi, jotka tapahtuvat NAND:n arkkitehtuurin ja piirin rakentamiseen käytettyjen aineiden fyysisten ominaisuuksien takia. Myöhemmin kerrotaan ECC:stä tarkemmin. (8, s. 1.)



KUVA 5. NAND:n käytön kasvaminen (9, s. 3)

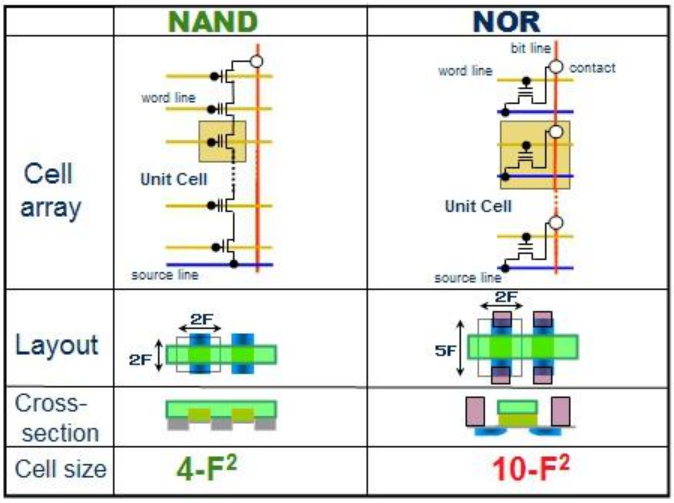


KUVA 6. Floating Gate -muistisolu (7, s. 20)

NAND-muistipiirien sivun koko on tavallisesti 2^n kB plus noin 64 kB vapaa-alue EEC:n informaatiota ja tukidataa varten. Sivut on järjestetty lohkoiksi 32 tai 64 sivua yhdessä lohkossa, eli 128 kB informaatiota per lohko. Pienin informaation määrä, jota voi kirjoittaa, on yksi tavu. Pienin tietojen määrä, jota onnistuu poistaa muistista, on yksi lohko, eli 128 kB (kilotavu). Riippuen piirin valmistajasta ja piirin mallista, yhtä sivua voidaan kirjoittaa osittain (partial programming) tietty lukumäärä kertoja tai ei ollenkaan. Datan poistaminen asettaa bittien arvoiksi ykköset, eli tyhjän muistipiirin luku palauttaa sarjaa 0xFF (vastaava desimaali luku on 256) luvuista. Datan kirjoittaminen muistipiirille muuttaa tarvittavien bittien arvot nollaksi. (10, s. 1–5.)

3.2 NAND- ja NOR-piirien vertailu

Ensimmäinen asia, joka täytyy huomata, on solun koko. NAND:n solu on vain $4F^2$ (Square Fermi), mikä on pienempi NOR:n solua ($10F^2$). Tähän vaikuttaa se fakta, että NOR:n solut tarvitsevat metallikontakteja niiden toimintaan, sillä solut on kytketty rinnakkain. NAND:n solut on kytketty sarjaksi, jotta on saatu poistettua ylimääräisiä kontakteja. Täten samankokoiselle NAND-piirille voi kirjoittaa enemmän informaatiota kuin vastaavankokoiselle NOR-piirille (johtuen piirin geometrisista parametreista). Kuvassa 7 on esitetty pääpiirteittäin NAND:n ja NOR:n solut. (11, s. 6–9.)

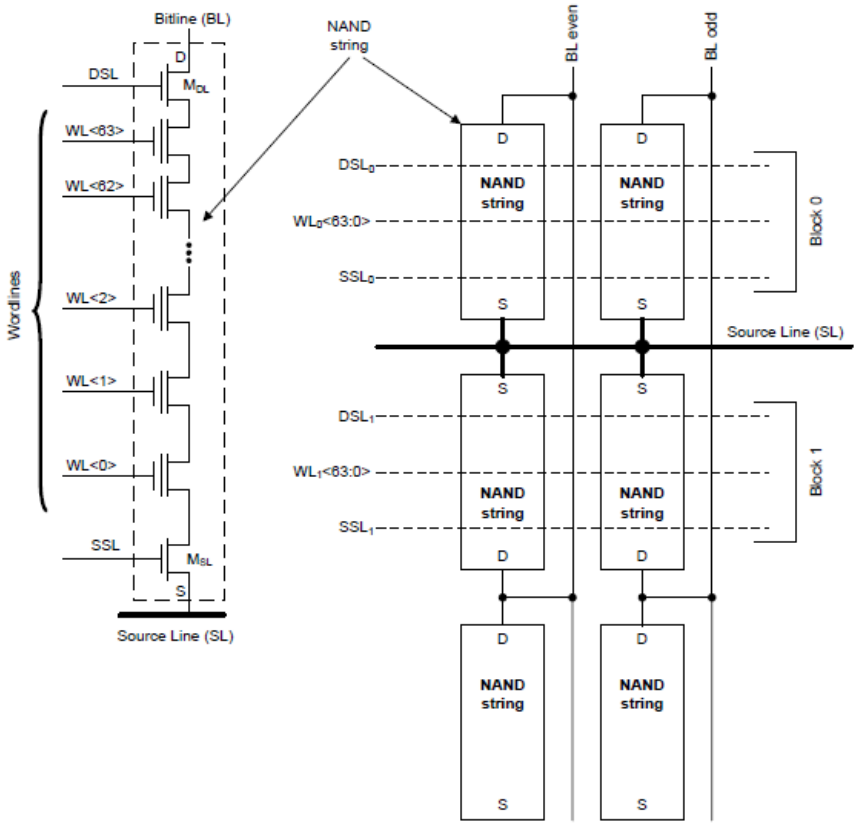


KUVA 7. NAND- ja NOR-piirien solujen vertailu (11, s. 6–9)

NAND:n etuina ovat tietojen nopea kirjoittaminen ja poistaminen (vain 2 ms vastaan NOR:n 750 ms). NOR:n etuna on satunnaisen datan saatavuus, mikä mahdollistaa XIP-arkkitehtuurin käyttöä. NAND-muistipiirit ideaalisesti sopivat datavarastoiksi. NOR-piiri vaatii 41 I/O-pinniä 16-bittiselle laitteelle, kun taas NAND vaatii 24 I/O-pinniä. NAND:n pinnien säilyttäminen tulee multipleksatusta (multiplexed) komento-, osoite- ja dataväylästä, kun NOR-piiri vaatii erillisiä väyliä datalle ja komennoille. (11, s. 6–9.)

3.3 NAND-arkkitehtuuri ja osoittaminen

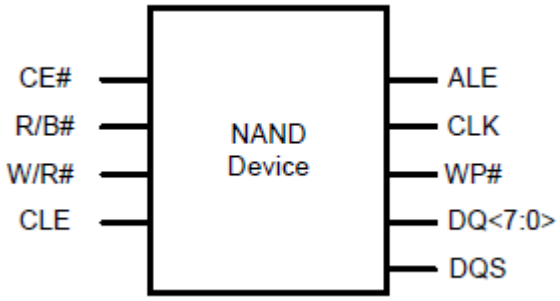
Muistipiirin koko on tosi tärkeä sulautetuissa järjestelmissä. Siksi muistipiirin solut on laitettu matriisimuotoiseksi optimoiden käytettyä paikkaa piissä. Kuten aikaisemmin mainittu, solut on kytketty sarjaksi NAND:n ketjuissa (string). Jokaisessa ketjussa voi olla 32 tai 64 solua (kuva 8). Kaksi valintatransistoria on sijoitettu ketjun reunoihin (edge) varmistamaan kytkentää lähdelinjaan (source line) M_{SL}:n kautta, ja bittilinjaan (bitline) M_{DL}:n kautta. Bittilinjat on jaettu muiden NAND:n ketjujen välillä. Ohjausportit (Control Gates) on kytketty word-linjojen kautta (WL_S). (7, s. 21.)



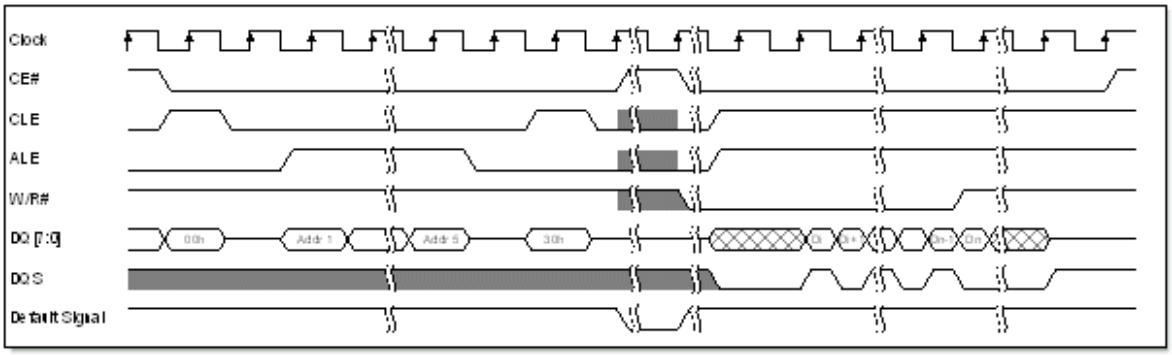
KUVA 8. NAND-ketju (vasemmalla) ja NAND-matriisi (oikealla) (7, s. 21)

Samassa word-linjassa olevia soluja on yhdistetty loogiseen sivuun. Sivujen lukumäärä on tyypillisesti 32 tai 64 kappaletta. Riippuen siitä, kuinka monta bittiä voi säilyttää yhdessä solussa, olemassa ovat seuraavat arkkitehtuurit: SLC (1 bitti per solu), MLC (2 bittiä per solu), 8LC (3 bittiä solussa) ja 16LC (4 bittiä per solu). Tässä tutkimuksessa käsitellään vain SLC-muistipiirejä, mutta kuitenkin jälkeenpäin mainitaan MLC:n eduista ja haitoista verrattuna SLC-muistipiiriin. Kannattaa sanoa, että kaikki NAND:n ketjut (strings), jotka kuuluvat word-linjojen tekemään ryhmään on tyhjennetty samanaikaisesti tietojen poistamisen operaation aikana. (7, s. 22.)

NAND:lla, kuten jokaisella muistipiirillä, on myös rajapinta, jonka kautta tapahtuu kommunikaatio ulospäin. On kaksi rajapinnan tyyppiä: synkroninen ja asynkroninen. ONFI 2.0 -standardissa (Open Flash NAND Interface) on ehdotettu ensimmäinen rajapinta, jossa luku- ja kirjoitusoperaatiot on yhdistetty yhteen signaaliin. Verrattuna asynkroniseen rajapintaan voi huomata, että muistipiiriin on liitetty yksi lisäsignaali, bidiractional strobe -signaali (DQS) (kuva 9). Datan siirtäminen tapahtuu DQS-signaalin molemmilla reunoilla (kuva 10). (12, s. 1.)

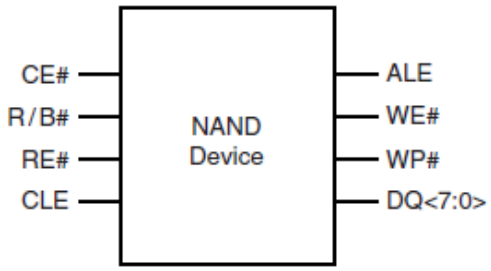


KUVA 9. SSI:ä (Source Synchronous Interface) tukevan NAND-muistipiirin pinnit (7, s. 27)



KUVA 10. Lukuoperaatio SSI:ssä (12, s. 1)

Asynkronisessa rajapinnassa (on käytössä tässä tutkimuksessa) luku- ja kirjoitussignaalit ovat erillisiä signaaleja CS- (Chip Select), command- ja address-salpojen (latch) mukana. Dataväylä voi olla 8- tai 16-bittinen. Tietoja siirrettäessä väylä on kokonaan käytössä, mutta komentojen lähettämisen aikana käytetään vain 8 bittiä. Kuvassa 11 on esitetty asynkroninen rajapinta. (12, s. 1.)



KUVA 11. Asynkroninen rajapinta (7, s. 27)

Luvun 3.1 loppuosassa on kerrottu lyhyesti muistipiirin loogisesta järjestyksestä, siksi tässä luvussa kerrotaan vain lyhyesti siitä, miten osoittaminen tapahtuu. Ennen komennon suorittamista on lähetettävä osoite, jossa pitää tehdä muutoksia tai josta saadaan informaatiota. Osoite on

jaettu rivi- ja sarakeosoitteiksi. Riviosoitteella löydetään tarvittava lohko ja sivu. Sarakeosoitteella löydetään osoitettava tavu sivussa. Ensiksi lähetetään sarakeosoite, sitten riviosoite. Yhdessä osoitesyklissä on 8 bittiä (address cycle). Ensimmäisessä syklissä siirretään LSB (Least Significant Bit). Osoitesykliä ei saa jakaa rivi- ja sarakeosoitteiden välille. (7, s. 27.)

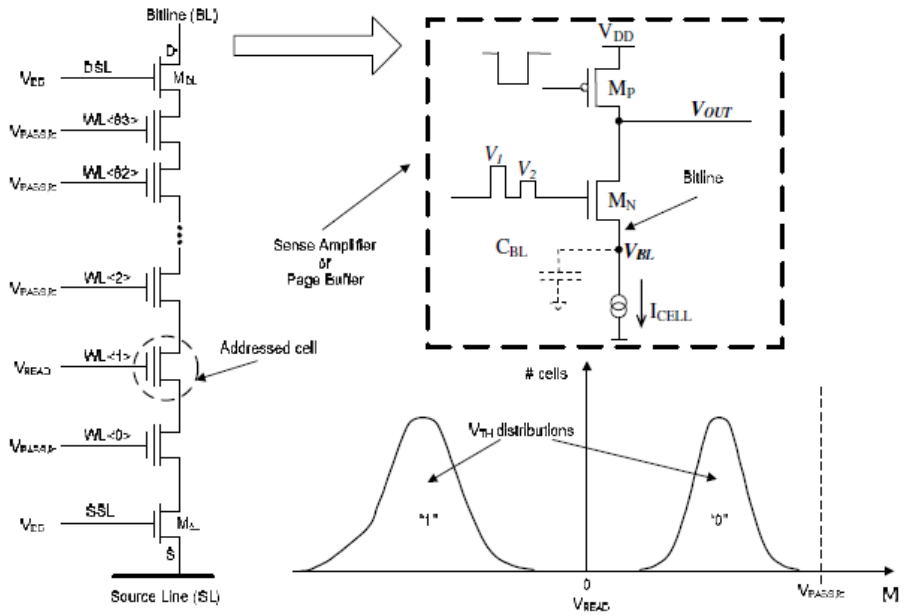
3.4 Perusoperaatiot

Tässä luvussa pyritään kuvaamaan perusoperaatioiden fyysiset ja loogiset aspektit. Sen tavoitteen saavuttamiseksi kuvataan lyhyesti Floating Gate -tekniikkaa. Floating Gate -muistisolu on esitetty kuvassa 6.

MOS (Metal Oxide Semiconductor) transistori sisältää kaksi päällekkäistä porttia: toisen portin pinta on kokonaisesti peitetty oksidilla ja toinen portti on kytketty muodostamaan porttiterminaalia. Eristetyn portin roolina on elektronien varasto, joka takaa elektronien säilyttämistä monien vuosien kuluessa. Elektronien injektointi ja vienti pois vastaavat kirjoitus- ja poisto-operaatiot. Nämä operaatiot muuttavat muistisolun kynnysjännitettä. Käyttämällä tiettyä jännitettä terminaalin portissa on mahdollista kertoa, mikä arvo muistisolussa on. Jos portin jännite on suurempi kuin kynnysjännite (V_H), siinä tapauksessa solussa on 1 eli solu on päällä. Jos V_H on suurempi portin jännitettä, solussa on 0, joten solu on pois päältä. (7, s. 20; 13, s. 1.)

3.4.1 Lukeminen

Solun lukemisen prosessin aikana sen ohjausportti (terminaaliportti) ajautuu V_{READ}-jännitteeseen (0 V). Muiden solujen porteille on asennettu V_{PASS}-jännite (tavallisesti 4–5 V) sillä tavalla, että ne pystyvät toimimaan johtimina huolimatta niiden kynnysjännitteistä. Tyhjennettyjen solujen kynnysjännite (V_H) on pienempi kuin 0 V, kirjoitettujen solujen V_H on suurempi kuin 0 V ja pienempi kuin 4 V. Käytännössä jos valitun solun portille asetetaan 0 V pääsyjännitteeksi (pass voltage), siinä tapauksessa solujen ketju johtaa virtaa vain silloin, kun osoitettu solu on tyhjennetty. Kuvassa 12 on esitetty pääsyjännitteen asettaminen lukuoperaation suorittamisessa. (7, s. 23.)



KUVA 12. Lukuoperaatio (7, s. 23)

Ketjun virta on tavallisesti 100–200 nA. Lukutekniikka käyttää kuvassa 12 olevaa loiskondensaattoria, C_{BL}. Kondensaattori on ladattu valmiiksi (precharged) kiinteäksi arvoksi, 1–1,2 V. Kun muistisolu on tyhjennetty, kondensaattori on purettu. Bittilinjan lataamisen aikana PMOS-transistori (M_P) on maadoitettu ja NMOS-transistori (M_N) pidetään kiinteällä jännitteellä, V_1 (noin 2 V). Lataamisen päättyessä bittilinjan jännite, V_{BL}, voi laskea kaavan 1 mukaisesti. (7, s. 23.)

$$V_{BL} = V_1 - V_{THN} \quad (1),$$

missä

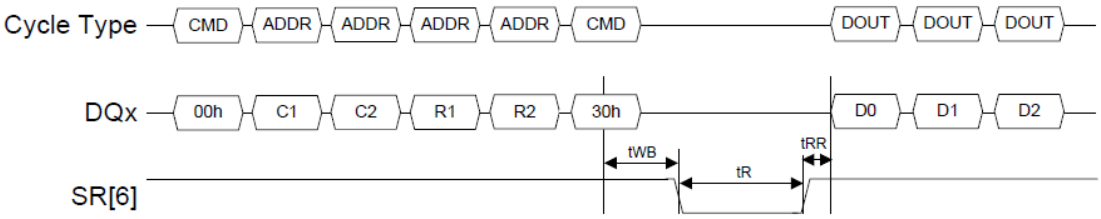
V_{THN} on NMOS:n kynnysjännite.

Transistorit M_N ja M_P on laitettu pois päältä, ja C_{BL} voi alkaa purkautua. Ajan T_{VAL} päästä, M_N-transistorin portti on asennettu V_2:ksi (1,4–1,6 V). Mainitun ajan pätkän jälkeen bittilinjan jännite voidaan laskea kaavan 2 mukaisesti. (7, s. 23.)

$$V_{BL} < V_2 - V_{THN} \quad (2)$$

M_N menee päälle ja noodin jännite V_{OUT} tulee yhtä suureksi kuin bittilinjan jännite. Tämän jälkeen V_{OUT} muunnetaan digitaaliseksi käyttämällä salpoja. (7, s. 23.)

Loogista lukuoperaatiota suoritetaan käyttämällä lukufunktioita, joiden avulla saadaan sivun dataa ulos. Dataa luetaan sivun rekisteristä alkaen sarakeosoitteesta. Lukeminen sivun osoiteavaruuden ulkopuolelta palauttaa määrittämätöntä tulosta. Kuvassa 13 on esitetty lukuoperaation algoritmi.



KUVA 13. Lukuoperaatio (14, s. 189)

ONFI-standardin mukainen komento ”0x30h” lähetetään ja odotetaan datan siirtämistä sivun rekisteriin (tR). Datan luku sivun rekisterin määritetystä sarakkeesta alkaa tR-ajan kuluessa. (14, s. 189.)

3.4.2 Kirjoittaminen (Program)

Kirjoitusoperaatio käyttää elektronin tunneloinnin kvanttiefektiä voimakkaassa sähkökentässä (Fowler–Nordheim-tunnelointi). Toisin sanoen, elektronit pääsevät eristeeseen metallista (terminaaliportista varastoon). Fowler–Nordheim-tunneloinnin kuvaus jätetään tästä tutkimuksesta pois. Käytännössä sähkökentän polaarisuudesta riippuen on suoritettu joko kirjoitus- tai poisto-operaatio. (7, s. 24.)

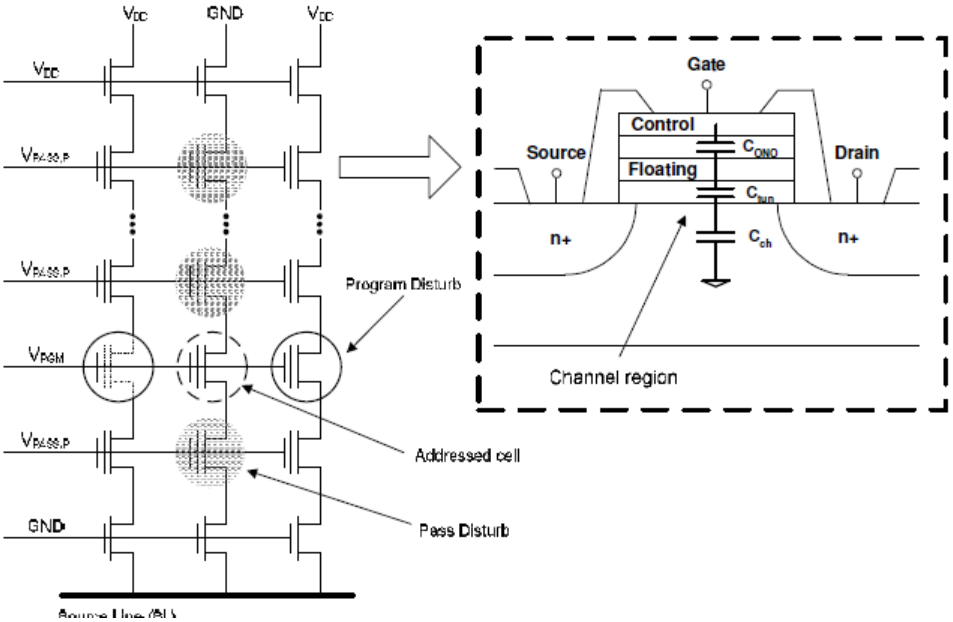
Täytyy myös mainita, että mitä voimakkaampi sähkökenttä on, sitä todennäköisemmin elektroni pääsee varastoon. Jotta saadaan voimakasta sähkökenttää, käytetään korkeaa jännitettä, joka on tämän menetelmän suuri haitta, sillä se vaikuttaa oksidin heikentymiseen. Metodin suurin etu on pieni käyttövirta, joka mahdollistaa muistisolujen rinnakkaiskirjoituksen. (7, s. 24.)

Elektronien päästämiseen varastoon (Floating Gate) käytetään seuraavia jännitteitä:

- V_{DD} lähtövalitsimen portilla
- V_{PGM} (20–25 V) valitulla portilla
- $V_{PASS,\ P}$ (8–10 V) muilla porteilla
- GND lähdevalitsimen portilla
- GND ohjelmoittavilla bittilinjoilla

- V_{DD} muilla bittilinjoilla (7, s. 24).

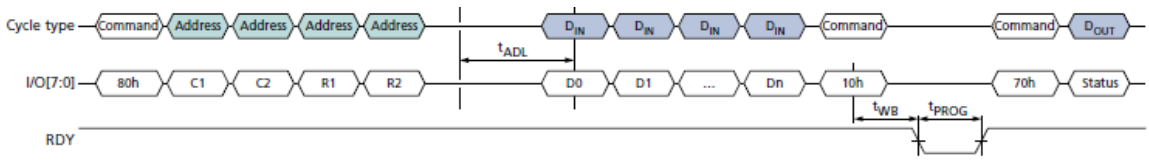
Menetelmän perusideana on korkean potentiaalin käyttäminen valitun solun loiskondensaattorin kautta suurentamaan potentiaalia oksidin alla (kuva 14) (7, s. 24).



KUVA 14. Fyysinen kirjoitusoperaatio (7, s. 24)

Asetettaessa pääsyjännitettä word-linjoille linjojen loiskondensaattorit tehostavat kanavan potentiaalia vähentämällä jännitteen pudotusta ja täten estämällä tunnelointia. (7, s. 24.)

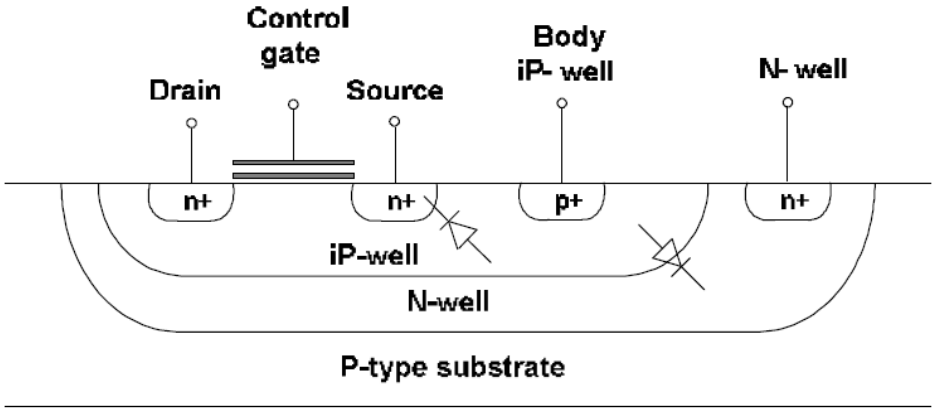
Loogisen kirjoitusoperaation aikana kopioidaan sivu tai sivun osa lähtien kohdasta, joka on määritetty sarakeosoitteella, sivurekisteriin. Tämän jälkeen kirjoitetaan sivurekisterin sisältöä muistipiirin matriisiin. Datan kirjoitus matriisiin vaatii tPROG-aikaa. Tämän jälkeen tarkastetaan kirjoituksen tila. Kirjoitusoperaation onnistuessa tilabitin arvo on nolla. Kuvassa 15 on esitetty looginen kirjoitusoperaatio. (14, s. 195.)



KUVA 15. Looginen kirjoitusoperaatio (14, s. 195)

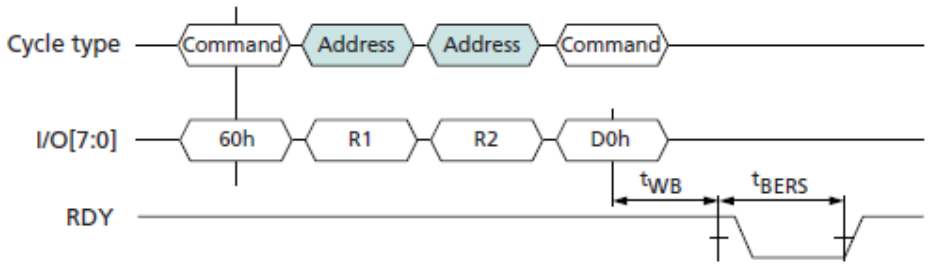
3.4.3 Poisto-operaatio (Erase)

NAND-muisti on sijoitettu triple-well-rakenteeseen ja tavallisesti jokainen taso (muistimatriisi) on erillisessä triple-well-rakenteessa (kuva 16). Tietojen poisto muistipiiristä saavutetaan vaikuttamalla iP-welliin korkealla jännitteellä ja maadoittamalla valitun lohkon word-linjat. Täten NAND ei vaadi negatiivista jännitettä datan poistamiseen. Saman tason muut lohkot myös käyttävät kyseessä olevaa iP-welliä ja niiden word-linjoja ei ole maadoitettu. Siten datan korruptointi on estetty. Tällä tavalla valitsemattomien word-linjojen potentiaali nousee ja Fowler–Nordheim-tunnelointi on estetty. Potentiaalin kasvaminen tapahtuu energian siirron vuoksi ohjausporttien ja iP-wellin välillä (capacitive coupling). Dataa poistettaessa käytetään hyvin korkeaa jännitettä. (7, s. 25.)



KUVA 16. Muistimatriisi triple-well-rakenteessa (7, s. 25)

Loogisen poisto-operaation aikana poistetaan data valitusta lohkosta. Lohko valitaan lähettämällä osoitesyklin (address cycle) aikana riviosoite. Poisto-operaation onnistuessa (tBERS-ajan päästä) tilabitin arvo on nolla. Kuvassa 17 on esitetty poisto-operaatio. (14, s. 182.)



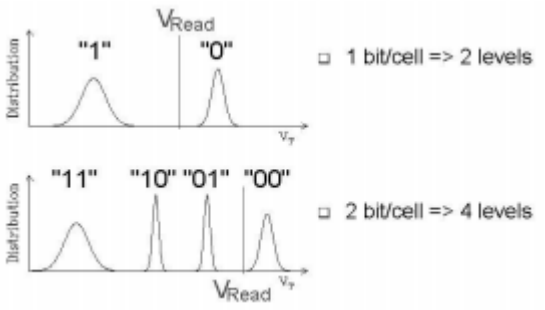
KUVA 17. Poisto-operaatio (14, s. 182)

3.5 SLC- ja MLC-piirien vertailu

Kuten aikaisemmin mainittiin, tässä tutkimuksessa käytettiin SLC-muistipiirejä. Siksi MLC:n kuvaus jätetään pois tästä tutkimuksesta. Seuraavaksi vain kerrotaan lyhyesti, mikä ero SLC:n ja MLC:n välillä on.

SLC-muistissa yhdessä muistisolussa on vain yksi bitti ja sen arvo on joko nolla tai yksi. MLC:ssä on kaksi bittiä per muistisolu. Täten MLC:n etuna on matriisin vaatima alue, koska samankokoisilla SLC:n ja MLC:n matriiseilla on eri tilavuusmahdollisuudet datan säilyttämiseksi. (15, s. 8.) SLC:ssä on parempi muistisolujen kestävyys, jolloin sillä voi suorittaa noin 100 000 kirjoitus- ja poistosykliä ennen muistisolun ”hajoamista”. MLC-muistipiirin solut kestävät vain 10 000 sykliä. (12, s. 1.) Lisäksi SLC-muistipiiri on nopeampi kuin MLC:

- MLC:ssä lukuoperaatio vaatii tehokkaampaa ECC-algoritmia (virheiden aste on korkeampaa) ja täten enemmän aikaa mahdollisten virheiden korjaukseen. Lisäksi dekoodaus algoritmin on tarkastettava neljä eri tilaa, mitä taas käyttää ylimääräistä aikaa (16, s. 1).
- MLC:ssä on käytössä samankokoinen kynnysjännitteen kehys kuin SLC:ssä (kuva 18), mutta on ladattava neljä eri varausta. Täten tarvitaan tehokkaampaa kirjoitusalgoritmia sekä pitempää aikaa takaamaan tarvittavaa tarkkuutta. (16, s. 1.)



KUVA 18. Kynnysjännitteen kehykset: SLC on ylempänä ja MLC on alempana (16, s. 1)

SLC-muistipiiri paremmin kestää hyvin korkeita ja hyvin matalia lämpötiloja. Siten edellä mainittuja muistipiirejä voidaan käyttää teollisuudessa (15, s. 8).

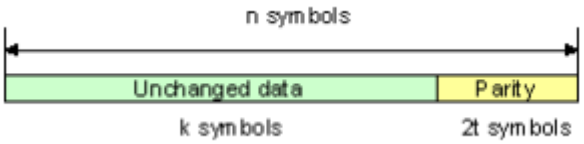
3.6 ECC (Error Correction Code)

NAND-muistissa floating gate -portilla oleva varaus voi muuttua sähköisesti sen korkean ja matalan arvojen välillä ja kuvata tällä tavalla loogisia ykkösiä ja nollia. Floating Gate -tekniikassa ei ole datan säilyttämisongelmia, vaan kirjoitus- ja poistosyklien suorittaminen vaikuttaa

negatiivisesti muistisolujen floating gate -portin oksidiin. Lisäksi floating gate -portti saa elektroneja lukemisen aikana, kun ohjausporttia pidetään jännitteessä V_{CC}. Varauksen purkaminen ja sen lisääminen voi vaikuttaa satunnaisbittivirheiksi. Flash-muistijärjestelmien on taattava datan säilyttäminen noin 10 vuoden aikana jopa 10^5 tai 10^6 kirjoitus- ja poistosyklin jälkeen. ECC:tä käytetään muistijärjestelmien luotettavuuden parantamiseksi. (17, s. 1.)

ECC-algoritmit, jotka korjaavat enemmän kuin yhden virheen, ovat Reed–Solomon- ja BCH-algoritmit (tutkijoiden nimistä Bose, Ray-Chaudhuri ja Hocquenghem). Virheiden korjaus suoritetaan lukuoperaation aikana. ECC lisätään dataan kirjoitusoperaation aikana. Riippuen ECC-skeemoista tarvitaan pariteettibittien eri lukumäärä. BCH- ja Reed–Solomon-algoritmeilla on samanlaiset struktuurit, mutta BCH-algoritmi vaatii pariteettibittien pienempää lukumäärää. ECC-algoritmi voi olla valmiiksi sisään rakennettu tai algoritmi toteutetaan erikseen ajurien mukana. (7, s. 43.)

Reed–Solomon-koodi (RS) on järjestelmällinen lineaarinen lohkokoodi. Tämä on lohkokoodi, koska data jaetaan lohkoiksi. Jokainen lohko jaetaan m-bittisiksi symboleiksi (tavallisesti 3–8 bittiä). Koodin lineaarisuus takaa sen, että jokainen m-bit -sana on validi symboli. ”Järjestelmällinen” tarkoittaa sitä, että dekoodattu data koostuu alkuperäisestä datasta ja liitetyistä pariteettibiteistä. RS voi korjata sekä sanoja, joissa on vain yksi virheellinen bitti, että sanoja, joissa kaikki bitit ovat virheellisiä. RS korjaa *(n - k) / 2* symboleja, missä *n - k* on pariteettibittien lukumäärä (kuva 19). (18, s. 1–2.)

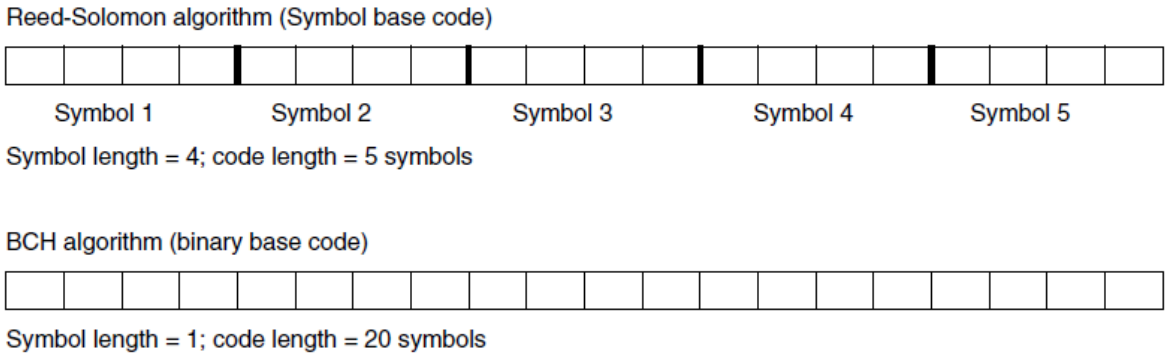


KUVA 19. RS-koodi (18, s. 1–2)

Hamming-koodin HW- tai SW-toteutus on suunnilleen yksinkertainen, mutta mainittu koodi voi korjata vain yhden virheellisen bitin. RS-algoritmi on tehokas, mutta vaatii paljon järjestelmän resursseja. BCH-algoritmi pystyy löytämään sekä useampia hajallaan olevia virheitä että hyvin keskitettyjä virheitä. BCH:n toinen etu on helpommat koodaus- ja dekoodausmenetelmät verrattuna RS-algoritmiin. BCH-koodi on lineaarinen lohkokoodi. (19, s. 3.)

Lohkokoodi koostuu vektoreista, jotka sisältävät N-elementtejä. Vektoreita kutsutaan sanoiksi, ja *N* on sanan pituus; *q* – sanan elementtejä. Lohkokoodi mappaa k-elementtejä N-pituiseen sanaan

ja suhde (ratio) $r = k / N$ on koodin aste (rate). Koodit on rakennettu kentistä, jotka koostuvat q-elementeistä. BCH-algoritmi on binaaripohjainen, kun RS-algoritmi symbolipohjainen. Kuvassa 20 on esitetty BCH-algoritmi verrattuna RS-algoritmiin. (19, s. 3.)

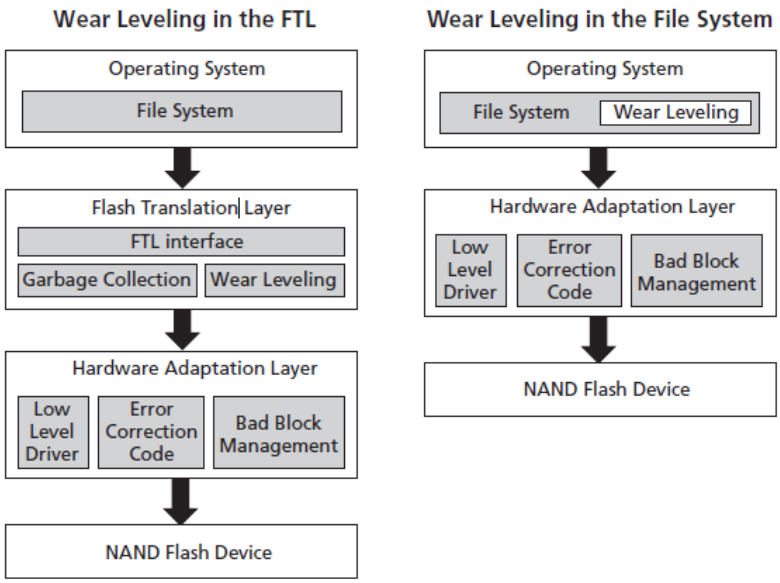


KUVA 20. RS ja BCH (19, s. 3)

3.7 Wear-leveling

SLC- ja MLC-muistipiireissä jokainen lohko kestää 10^5 ja 10^4 kirjoitus- ja poistosykliä vastaavasti. Jos sovellus useasti suorittaa kirjoitus- ja poistosyklejä, niin on suositeltu toteuttaa wear-leveling -algoritmi. Tämä algoritmi seuraa jokaisen lohkon mainittujen syklien lukumäärää. Muistijärjestelmissä, joissa ei ole käytössä wear-leveling -algoritmia, joitakin lohkoja käytetään useammin. Wear-leveling -algoritmi varmistaa sen, että jokaisella lohkolla on suunnilleen sama kirjoitus- ja poistosyklien lukumäärä. (20, s. 1–2.)

Wear-leveling -algoritmi toteutetaan tavallisesti FTL:n (Flash Translation Layer) sisällä. FTL on ohjelmistotaso, joka sijaitsee tiedostojärjestelmän ja NAND-muistin välillä. FTL tarjoaa käyttöjärjestelmälle rajapintaa, jota käyttämällä käyttöjärjestelmä voi kirjoittaa tai lukea dataa NAND:in tai NAND:stä samalla tavalla kuin kovalevyn ajurit tekevät. FTL muuntaa virtuaaliset osoitteet fyysisiksi. Wear-leveling -algoritmi voidaan toteuttaa NAND-muistin tiedostojärjestelmän osana (kuva 21). (20, s. 1–2.)



KUVA 21. FTL (20, s. 2)

Järjestelmissä, jotka käyttävät FAT-tiedostojärjestelmää (File Allocation Table), tiedostojärjestelmä sijaitsee samoissa virtuaalilohkoissa. FAT-taulun usein tapahtuvat päivitykset vaativat kirjoitusoperaatioiden suoritusta, jotka olettavat datan poistoa samasta fyysisestä lohkosta. Täten muistipiirin elinaika lyhenee hyvin nopeasti. Esimerkiksi oletetaan, että on olemassa järjestelmä, joka käyttää FAT32-tiedostojärjestelmää. Tiedostojärjestelmän yhden sektorin koko on 2 kB. Lasketaan se, kuinka monta kertaa päivitetään FAT-taulua kirjoittamalla 10 MB -kokoisen tiedoston NAND-muistiin, jolla yhden lohkon koko on 16 kB. Kirjoittaessa 10 MB -kokoisen tiedoston, tiedostojärjestelmä vaatii 5 kB -kokoisia kirjauksia (entries) ja 5 kB -kokoisia klustereita. Tätä varten tarvitaan 640 fyysistä lohkoa. Tämä tarkoittaa sitä, että järjestelmä onnistuu kirjoittaa dataa samaan paikkaan 20 kertaa (20 * 5 120 = 102 400). (20, s. 2–3.)

NAND-muistipiirin elinaika lasketaan kaavan 3 mukaisesti:

$$Elinaika = \frac{Muistin\ koko * poistosyklien\ lukumäärä * FAT\text{-}kerroin}{Kirjoitettujen\ tavujen\ lukumäärä\ per\ päivä} \quad (3)$$

Oletetaan, että kirjoitusnopeus on 3 kB/s:

$$Elinaika = \frac{10\ MB * 20 * 0{,}7}{(3\ KB/s) * 24 * 60 * 60} = 0{,}55\ päivää$$

NAND-muistissa, jossa virtuaalilohkot on mapattu samoihin fyysisiin lohkoihin, muistipiirin elinaika vähenee kovasti riippumatta muistin koosta. (20, s. 2–3.)

Kuten oli mainittu aikaisemmin, wear-leveling -algoritmi pidentää muistin elinaikaa. Virtuaalilohko mapataan toiseen fyysiseen lohkoon jokaisen kirjoitus- ja poistosyklin jälkeen varmistaen, että kirjoitus- ja poistosyklejä on sama lukumäärä jokaiselle lohkolle. (20, s. 2–3.) Kaavan 3 mukaisesti muistipiirin elinaika on:

$$Elinaika = \frac{64\ MB * 100 * 0{,}7}{(3\ kB/s) * 24 * 60 * 60} = 18124\ päivää\ noin\ 49{,}7\ vuotta$$

Olemassa on kaksi wear-leveling -algoritmin tyyppiä, jota voi toteuttaa FTL:ssa:

- Dynaaminen wear-leveling -algoritmi
- Staattinen wear-leveling -algoritmi (20, s. 2–3.)

Dynaamista algoritmia sovellettaessa uusi data tallennetaan vapaille lohkoille, joilla on kirjoitus- ja poistosyklien minimaalinen lukumäärä (20, s. 2–3).

Staattinen wear-leveling -algoritmin on käytössä silloin, kun siirretään staattista dataa (esimerkiksi koodia) toiseen lohkoon. Täten vapautettua lohkoa voi käyttää datalle, jota on päivitetty hyvin useasti. Saavutettaessa tiettyjen kirjoitus- ja poistosyklien minimi- ja maksimilukumäärien välillä oleva ero, käynnistetään staattinen algoritmi. (20, s. 2–3.)

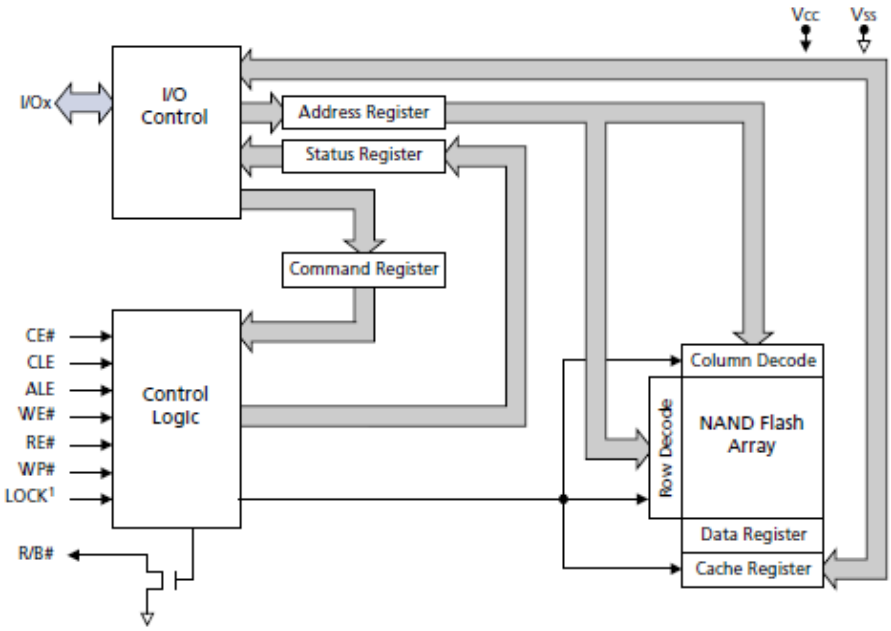
Käytettäessä wear-leveling -algoritmia on muistettava, että vapaiden lohkojen lukumäärä kerrasta toiseen pienenee. Joskus voi olla tilanne, kun ei ole tarpeeksi vapaita lohkoja. Siinä tapauksessa käynnistetään Garbage Collection -algoritmi. Garbage Collection -algoritmi auttaa estämään muistifragmentaatiota. Lisäksi kyseessä oleva algoritmi takaa, että järjestelmässä on aina tarpeeksi vapaita lohkoja datan kirjoittamiseksi. Algoritmi voidaan käynnistää säännöllisesti tai käyttöjärjestelmä suorittaa sen taustalla. (12, s. 1.)

4 KÄYTETTYJEN TOSHIBAN JA MICRONIN PIIRIEN KUVAUS

Tässä luvussa lyhyesti kuvataan tutkimuksen aikana käytettyjä muistipiirejä. Tarkempaa informaatiota voi löytää Toshiban ja Micronin ehdottamasta dokumentaatiosta. Kyseessä olevat muistipiirit ovat SLC-tyyppiset. Ensimmäinen lohko on oletuksena hyvä aina jokaisessa piirissä. Hyvien lohkojen minimilukumäärä on 1004. Toshiban muistipiiri on Micronin muistipiiriä pienempi ja sitä halvempi. Lisäksi Toshiban muistipiiriin on rakennettu ECC-ominaisuus. Micronin muistipiirille mainittu algoritmi toteutetaan erikseen. Tarkastettaessa lohkon tilaa luetaan spare-alueen ensimmäinen tavu. Lohko on hyvä, jos luetun tavun arvo on 0xFF. (21, s. 1, 76; 22, s. 1, 35.)

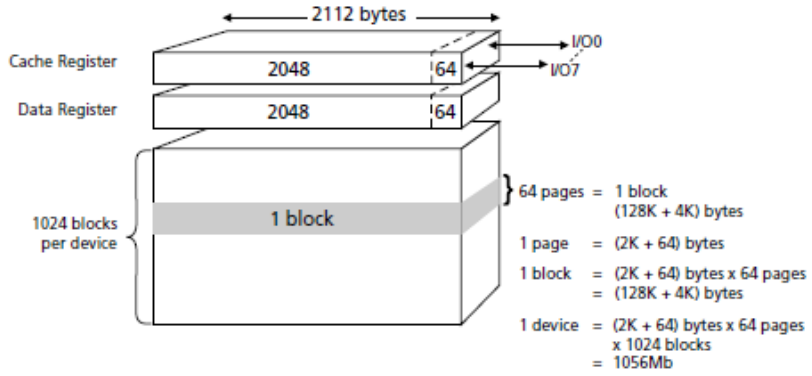
4.1 Micronin muistipiiri

Micronin muistipiiri käyttää asynkronista rajapintaa, joka mahdollistaa I/O-toimintojen nopean suorittamisen. Sitä paitsi käytössä on multipleksattu 8-bittinen väylä komentojen, osoitteiden ja datan lähettämistä ja vastaanottamista varten. Asynkronisella rajapinnalla on viisi ohjaussignaalia käytössä: CE# (Chip Enable), CLE (Command Latch Enable), ALE (Address Latch Enable), WE# (Write Enable) sekä RE# (Read Enable). Yllä mainitun lisäksi on kaksi ohjaussignaalia. WP# (Write Protection), jolla voi sallia tai kieltää kirjoitus- ja poistotoimintojen suorittamista. R/B#-signaali, jolla seurataan muistipiirin tilaa. Kyseessä olevan piirin HW-rajapinta ehdottaa standardoitujen pinnien listan, jotka ovat riippumattomia muistisolujen tiheydestä. Tämä mahdollistaa muistisolujen tiheyden kasvamista arkkitehtuuria muuttamatta. Muistipiirillä on yksi looginen yksikkö, joka suorittaa komentoja ja ilmoittaa piirin tilasta. Komentorekisteri vastaanottaa I/O-pinneistä tulevia komentoja ja lähettää niitä Control Logic -yksikköön, joka saadun signaalin perusteella generoi impulsseja muistipiirin ohjaamiseksi. Saatu osoite käsitellään osoiterekisterissä, eli dekoodataan rivi- ja sarakeosoitteet. Tiedot muistipiiristä lähetetään tavu kerrallaan. Samalla periaatteella vastaanotetaan dataa muistipiirissä, eli tavu kerrallaan. Kirjoitus- ja lukuoperaatiot ovat sivupohjaiset, poisto-operaatio on lohkopohjainen. Tilarekisteri ilmoittaa kiteen (die) operaation tilasta tietyssä ajan hetkessä. Kuvassa 22 on esitetty piirin toiminnallinen lohkokaavio. (21, s. 1–15.)



KUVA 22. Micronin muistipiirin toiminnallinen lohkokaavio (21, s. 1–15)

Muistipiirin matriisissa on 1024 lohkoa. Jokaisessa lohkossa on 64 sivua. Yhden sivun koko on 2112 tavua, josta 64 tavua käytetään spare-alueeksi. Kuvassa 23 on muistipiirin matriisi. (21, s. 16.)

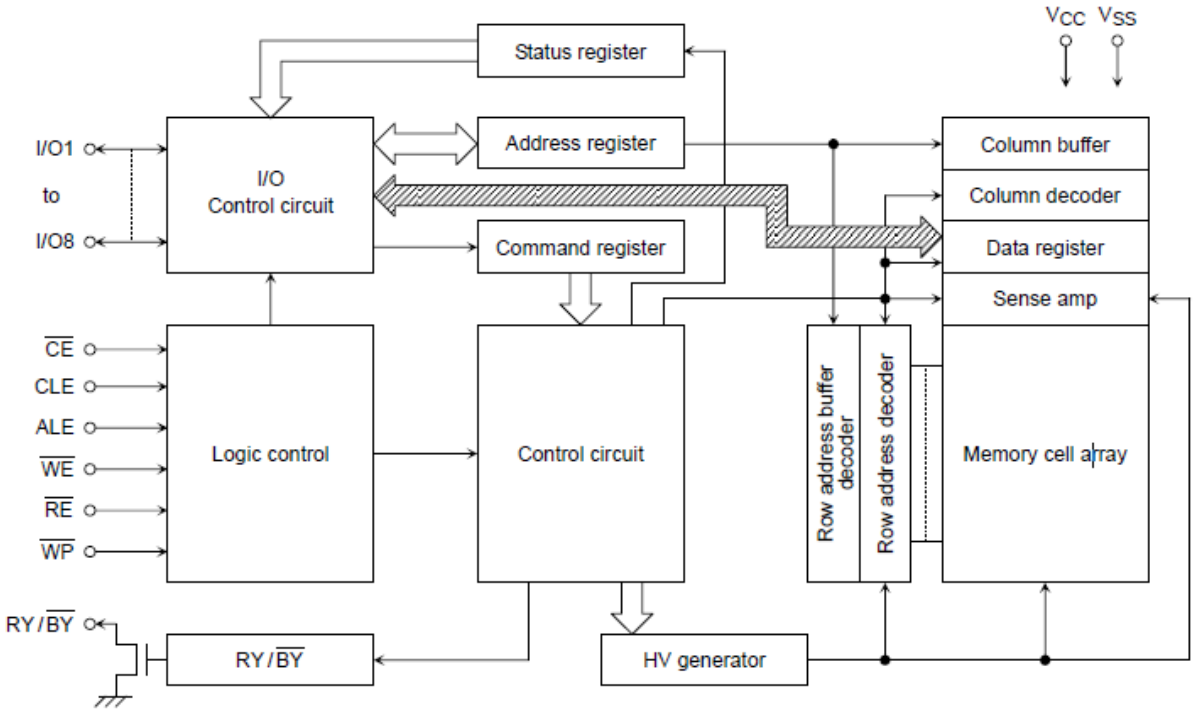


KUVA 23. Muistipiirin matriisi (21, s. 16)

4.2 Toshiban muistipiiri

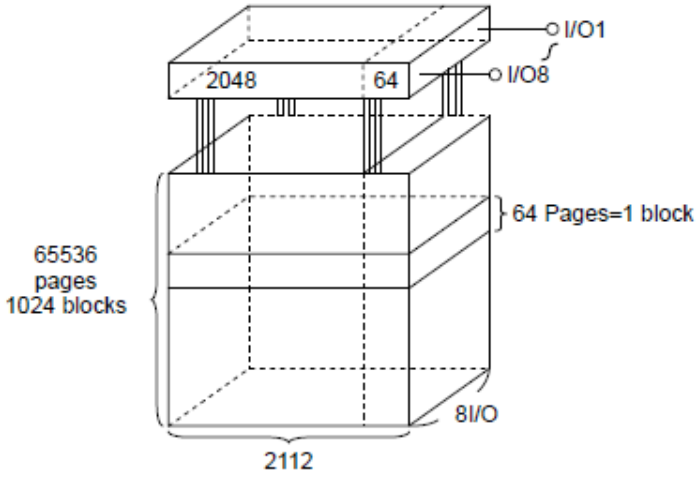
Toshiban muistipiirin toiminta ei eroa paljon Micronin muistipiirin toiminnasta. Toshibakin käyttää multipleksattua 8-bittistä väylää komentojen, osoitteiden sekä datan lähettämistä ja vastaanottamista varten. Kuten Micronin muistipiiri, Toshiban piiri myös käyttää Micronin kuvauksessa olevia ohjaussignaaleja. Kyseessä oleva piiri vastaanottaa ja lähettää dataa samalla periaatteella kuin Micronin valmistama piiri, eli tavu kerrallaan. Toshiban tekemän piirin toiminta

on samankaltainen kuin Micronilla. Komentorekisteri vastaanottaa I/O-pinneistä tulevia komentoja ja lähettää niitä Logic Control -yksikköön, joka saadun signaalin perusteella generoi impulsseja muistipiiriä ohjaamiseksi. Saatu osoite käsitellään osoiterekisterissä, eli dekoodataan rivi- ja sarakeosoitteet. Tätä voi nähdä toiminnallisesta lohkokaaviosta (kuva 24). Muistipiirissä myös on vain yksi looginen yksikkö (LUN, Logic Unit). (22, s. 1–17.)



KUVA 24. Toshiban muistipiirin toiminnallinen lohkokaavio (22, s. 1–17)

Kirjoitus- ja lukutoiminnat suoritetaan myös sivupohjaisesti, poistotoiminta on lohkopohjainen. Muistipiirissä on 1024 lohkoa. Yhdessä lohkossa on 64 sivua, joista hyvien sivujen minimilukumäärä on 1004. Sivun koko on 2112 tavua ja niistä 64 tavua on spare-alue. Kuvassa 25 on muistipiirin matriisi. (22, s. 18.)



KUVA 25. Muistipiirin matriisi (22, s. 18)

5 TESTITAPAUKSET

Testitapausten valinta perustuu muistipiirien dokumentaatioon ja dokumentaatiossa väitettyihin parametreihin, kuten sivun osittaisen kirjoitustoiminnan (partial programming) kertojen lukumäärään. Testauksen aikana pyrittiin kuitenkin vertaamaan Toshiban ja Micronin muistipiirejä keskenään. Lisäksi saatiin muistipiirien käytön tilasto, kun laitteen firmware on tallennettu piirille ja heti tallennuksen jälkeen (luku-, kirjoitus- ja poisto-operaatioiden lukumäärät sekä mapattuja virtuaalilohkoja). Seuraavissa luvuissa kuvataan lyhyesti testitapauksia ja niiden algoritmia.

5.1 Bad Blocks -testitapaus

Huono lohko on lohko, joka sisältää vähintään yhden kelvottoman (invalid) bitin. Piirien valmistaja suorittaa erilaisia testejä valmistuksen aikana käyttämällä äärilämpötiloja ja äärijännitteitä. Kaikki huonot lohkot, jotka ilmestyvät tehtaan testauksesta, merkitään tietyllä tavalla. Tavallisesti valmistaja ilmoittaa piirin dokumentaatiossa, miten voidaan erottaa huonoja lohkoja ja niiden maksimilukumäärän. Huonot lohkot myös voivat esiintyä muistipiirin käytön aikana. Huonojen lohkojen olemassaolo ei vaikuta mitenkään muistipiirin toimintaan, koska jokainen huono lohko on ”itsenäinen” ja yksilöllisesti eristetty bittilinjoista lohkon valitsintransistorilla. Huonojen lohkojen informaatiota on luettava ennen poistotoimintaa, sillä merkinnän voi pyyhkiä ja siten saada määrittelemättömiä virheitä tulevaisuudessa. Edellä mainittu toiminta tavallisesti toteutetaan NAND-muistipiirin ajureissa. (23, s. 1–2.)

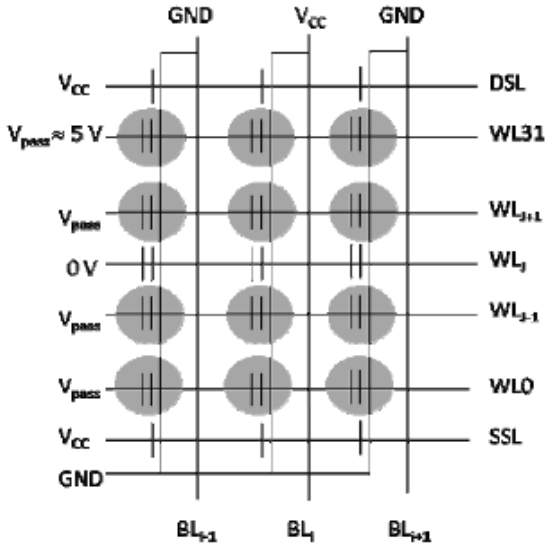
Tämän testitapauksen algoritmi on yksinkertainen:

1. Osoitetaan jokaisen lohkon ensimmäisen sivun 2048. tavuun.
2. Jos tavun arvo ei ole 0xFF, lohko on huono.
3. Lisätään huono lohko listaan, jos kyseessä oleva lohko löytyi.
4. Luetaan taulu läpi ja näytetään tulokset.

Yllä mainitun algoritmin avulla löydetään vain lohkot, jotka on merkitty tehtaalla testauksen aikana huonoiksi, mikä tässä tutkimuksessa oli tarpeeksi muiden testitapausten suorittamista varten. Tämä testitapaus suoritettiin aina ensimmäisenä. Testitapauksen tuloksia käytettiin muissakin testeissä. Saadut tulokset hyvissä ja huonoissa lohkoissa eroavat kovasti toisistaan. Virheen esiintymisen todennäköisyys on korkeampaa huonon lohkon käyttämisen tapauksessa.

5.2 Read Disturb -testitapaus

NAND-muistipiirien arkkitehtuurissa kyseessä oleva häiriö ilmestyy muita häiriöitä useammin. Luettaessa samaa muistisolua monta kertaa, kirjoittamatta uudestaan mainittua solua, voidaan saada esiintymään lukuhäiriötä. Jotta luetaan valitun muistisolun dataa, on laitettava samaan ketjuun kuuluvat muistisolut ON-tilaan riippumatta niiden säilyttämästä varauksesta. Mainittujen solujen ohjausportteihin asetettu suhteellisesti korkea jännite ja peräkkäisen lukuoperaation aikana käytetyn korkean jännitteen pulssit voivat vaikuttaa SILC-ilmiöön (Stress Induced Leakage Current), siten jotkin solut saavat ylimääräisiä elektroneja. Muistisolujen kynnysjännite muuttuu positiiviseen suuntaan. Osoitettaessa edellä mainittuja soluja voi aiheutua lukuvirheitä. SILC-ilmiö ei ole symmetrinen ja siten ongelmallisilla soluilla ei välttämättä ole varauksen säilyttämisongelmia. Lukuhäiriö aiheuttaa oksidin väliaikaisia vaurioita. Solun uudestaan kirjoittaminen korjaa lukuhäiriön tekemiä vaurioita. Tämän jälkeen muistisolun floating gate sisältää varauksen oikean arvon. Kuvassa 26 on esitetty lukuhäiriö. SILC-ilmiön kuvaus jätetään pois tutkimuksesta kokonaan. (7, s. 99.)



KUVA 26. Lukuhäiriö (harmaana on merkitty solut, joihin lukuhäiriö mahdollisesti voi vaikuttaa) (7, s. 99)

Testitapauksen algoritmi oli seuraava:

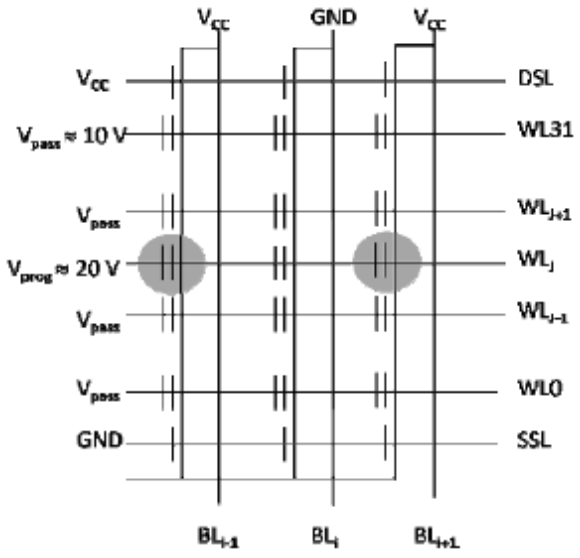
1. Valitaan lohko ja suoritetaan poisto-operaatio.
2. Valitaan lohkon sisällä sivu.

3. Kirjoitetaan sivuun tietty data-malli. Tutkimuksessa käytettiin näitä data-malleja: 0x55 ja 0xAA.
4. Luetaan sivu yhden kerran ja tulokset tallennetaan.
5. Käynnistetään lukualgoritmi ja luetaan valittu sivu 1 000 000 kertaa. Muistipiirien dokumentaatiossa tämä on suositeltu peräkkäisten lukuoperaatioiden lukumäärä (21, s. 1–15; 22, s. 1–17).
6. Luetaan valittu sivu uudestaan ja verrataan tulokset säilytettyyn dataan.
7. Otetaan talteen virheiden osoitteita.
8. Poistetaan data.

Yllä oleva algoritmi suoritettiin jokaisessa lohkossa, sekä huonoissa että hyvissä.

5.3 Program Disturb -testitapaus

Kirjoitushäiriö aiheuttaa virheitä, jos yhden tai useamman bitin arvo muuttuu toiseksi (ykkösestä nollaksi). Kirjoitusoperaation aikana käytetyt jännitteet aiheuttavat kirjoitushäiriötä. Korkea jännite vaikuttaa vain kirjoitettavaan muistisoluun. Muihin muistisoluihin vaikuttavat matalammat jännitteet. Siten voi tapahtua muistisolujen ”pehmeä ohjelmointi” (Soft Programming), eli muiden solujen tapauksessa myös voi syntyä tunnelointi-ilmiö. Kirjoitushäiriön virheet voivat esiintyä kirjoitettavassa sivussa tai muissa lohkon sisällä olevissa sivuissa. Löydettäessä häiriön aiheuttamia virheitä valitusta lohkosta pyyhitään data pois lohkosta ja kirjoitetaan data uudestaan lohkoon (24, s. 2). Kuvassa 27 on esitetty kirjoitushäiriö. (7, s. 100.)



KUVA 27. Kirjoitushäiriö (solut, joihin kirjoitushäiriö voi mahdollisesti vaikuttaa, on merkitty harmaalla) (7, s. 100)

Testitapauksessa käytettiin seuraavaa algoritmia:

1. Valitaan lohko ja poistetaan vanha data.
2. Tarkistetaan, että lohko sisältää vain tyhjän lohkon vastaavan data-mallin (0xFF).
3. Valitaan lohkon sisällä oleva sivu ja kirjoitetaan sinne data-malli, ensiksi 0xAA ja seuraavan kerran 0x55. Sivu kirjoitetaan osittain.
4. Luetaan lohkon ja tallennetaan virheiden osoitteita.
5. Poistetaan data lohkosta.

Testitapaus suoritettiin jokaisessa lohkossa.

5.4 Erase Disturb -testitapaus

Poisto-operaation synnyttämä häiriö ei ole iso ongelma NAND-muistipiireissä, koska pyyhitään data koko lohkosta. Kuitenkin datan poisto yhdestä lohkosta voi aiheuttaa tietojen osittaista poistoa muista lohkoista. Virheen ilmestymiseen voivat vaikuttaa vuotovirtoja laskemalla word-linjojen potentiaalia ja siten mahdollistaa tunnelointi-ilmiötä. (7, s. 77.)

Testitapaus suoritettiin sekä huonoissa että hyvissä lohkoissa. Testitapauksissa käytettiin seuraava algoritmi:

1. Valitaan lohko ja poistetaan data sekä valitusta lohkosta, että naapurilohkoista.
2. Kirjoitetaan datan malli valittuun lohkoon ja naapurilohkoihin. Data-mallit ovat samanlaisia kuten aikaisemmin mainituissa testitapauksissa (0xAA ja 0x55).
3. Poistetaan data valitusta lohkosta.
4. Luetaan naapurilohkot ja tallennetaan virheiden osoitteet.
5. Poistetaan data naapurilohkoista.

5.5 Program/Erase -testitapaus

NAND-piireissä on rajoitettu elinaika. SLC-muistipiireillä, kuten aikaisemmin mainittu, elinaika on noin 100 000 kirjoitus- ja poistosykliä. Kirjoitus- ja poisto-operaatioiden aikana muistisolut saavat ylimääräistä varausta, mikä aiheuttaa oksidin heikentymistä, eli elektronit jäivät oksidiin. Ylimääräisen varauksen saaminen aiheuttaa kirjoitus- ja poisto-operaatioiden suoritusajan kasvua. Edellä mainitun suoritusajan kasvu voi aiheuttaa sisäisen ajastimen ylivuotoa ja siten kirjoitus- tai poisto-operaatio epäonnistuu. (24, s. 1.)

Oksidin heikentymistä ei voida korjata poisto-operaation avulla. Kuitenkin NAND-muistipiirin elinaikaa voi pitkittää luvussa 3.7 mainitun wear-leveling -algoritmin avulla. (24, s. 1.)

Kyseessä oleva testitapaus suoritettiin vain hyvissä lohkoissa. Algoritmi on suhteellisesti yksinkertainen:

1. Valitaan lohko ja poistetaan vanha data. Tämä poisto-operaatio otetaan huomioon käsiteltäessä tuloksia.
2. Suoritetaan kirjoitus- ja poistosyklit.
3. Kirjoitus- ja poisto-operaatioiden jälkeen tarkastetaan operaation tulos.
4. Jos operaatio epäonnistuu, keskeytetään testitapaus.
5. Luetaan lohko ja virheellisen datan osoitteet otetaan talteen.

5.6 Partial Page Programming -testitapaus

Kuten aikaisemmin oli kuvattu, kirjoitusoperaatio voi aiheuttaa virheitä ja muiden kuin valitun solun muistisolujen Soft Programming -ilmiötä. Siten dokumentaatiossa on hyvin rajoitettu sivun osittaisen kirjoitusoperaatioiden lukumäärää; sekä Toshiban että Micronin muistipiirit kestävät vain neljä mainittua operaatiota. (21, s. 84; 22, s. 6.) Tämä on otettava huomioon tiedostojärjestelmän toteuttamisen aikana.

Renesas Mobilen suunnitteleman laitteen toiminnan näkökulmasta on todella tärkeää, että sivun osittainen kirjoitusoperaatio ei aiheuttaisi mitään virheitä neljän syklin sisällä, sillä käytetty tiedostojärjestelmä jakaa jokaisen sivun neljäksi sektoriksi. Testitapauksen sivutavoitteena oli selvittää, kuinka monta kyseessä olevaa operaatiota muistipiiri voi kestää ennen ensimmäisten virheiden ilmestymistä.

Testitapaus suoritettiin sekä hyvissä että huonoissa lohkoissa. Testissä käytettiin 2 data-mallia (0xAA ja 0x55). Testitapauksen algoritmi oli seuraava:

1. Valitaan lohko ja poistetaan vanha data.
2. Valitaan sivu ja kirjoitetaan data-malli siihen osittain neljä kertaa.
3. Jokaisen kirjoitusoperaation jälkeen tarkastetaan lohkon sisältöä käyttämällä lukuoperaatiota.
4. Jos virheitä löytyy, tallennetaan niiden osoitteet. Toisessa tapauksessa aloitetaan testitapauksen toinen vaihe, eli selvitetään operaatioiden maksimilukumäärä.

5. Kirjoitusoperaation jälkeen suoritetaan lukuoperaatio. Jos virheitä löytyy, keskeytetään testitapaus ja otetaan virheiden osoitteet talteen.

5.7 Data Retention -testitapaus

Data Retention on NAND-muistipiirin kyky säilyttää data muuttamattomana tietyn aikakauden sisällä. Lämpötilan kasvaessa muistipiirin Data Retention -ominaisuus huononee. Täten käyttämällä Arrheniusin yhtälöä (kaava 4), voidaan laskea testitapauksen vaatima aika ja lämpötila. (26, s. 3.) Tutkimuksessa käytössä olevien muistipiirien dokumentaatiosta löydetään dokumentoitu aika (10 vuotta). Muistipiirien valmistajat (Toshiba ja Micron) takaavat, että tämän aikakauden sisällä piireissä oleva data ei muutu miksikään. (21, s. 1; 25, s. 27.)

$$AF = e^{-\frac{E_a}{k} \cdot \left(\frac{1}{T_2} - \frac{1}{T_1}\right)} \quad (4),$$

missä

AF on nopeutuskerroin,

E_a on aktivaatioenergia,

k on Boltzmannin vakio ($8{,}623 \cdot 10^{-5}$ eV/K),

T_1 on sovelluksen käyttöympäristön lämpötila Kelvineissä ja

T_2 *on* akseleroitu rasituslämpötila Kelvineissä

Kaavan 5 avulla voidaan laskea muistipiirin datan säilymisaika (data retention) (26, s. 4).

$$Säilymisaika = \frac{uunissa\ pitoaika * AF}{tuntien\ lukumäärä\ vuorokaudessa * päivien\ lukumäärä\ vuodessa} \quad (5)$$

Arrheniusin teorian avulla voi testata mitä tahansa laitetta nopeutetussa ympäristössä sekä ennakoida laitteen käyttäytymistä käyttöympäristössä pitkään.

NAND-muistipiiri ei ole protolaitteesta irrotettavaa. Protolaitteessa on olemassa muut piirit ja komponentit, joille korkealämpötila voi aiheuttaa vaurioita. Siksi valitaan 100 °C testilämpötilaksi T_2. Aktivaatioenergian arvon löytyy IEEE-dokumentaatiosta ja sen arvo on 1,0 eV (27, s. 1). Lämpötila T_1 on laitteen käyttölämpötila, täten T_1 on noin 25 °C. Sijoitetaan lämpötilat kaavaan 4:

$$AF = e^{\frac{E_a}{k} \cdot \left(\frac{1}{T_2} - \frac{1}{T_1}\right)} = e^{\frac{-1{,}0}{8{,}623 \cdot 10^{-5}} \cdot \left(\frac{1}{(100+273{,}15)} - \frac{1}{(25+273{,}15)}\right)} \approx 2485$$

Kaavasta 5 johtuu uunissa pitoaika:

$$uunissapitoaika = \frac{Säilymisaika * tuntien\ lukumäärä\ vuorokaudessa * päivien\ lukumäärä\ vuodessa}{AF} \quad (6)$$

Sijoitetaan nopeutuskerroin kaavaan 6:

$$uunissapitoaika = \frac{10 * 24 * 365}{2485} = 35\ h\ eli\ 1,5\ päivää$$

Yllä olevasta laskelmasta käytettiin uunissapitoaikaa testitapauksessa. Kyseessä oleva testitapaus suoritettiin kaikkien testien jälkeen. Algoritmi oli seuraava:

1. Pyyhitään hyvät lohkot.
2. Kirjoitetaan data-malli (0xAA) hyville lohkoille.
3. Luetaan data lohkoista ja tarkastetaan
4. Jos on virheitä, merkitään virheiden osoitteet.
5. Tallennetaan tiedot.
6. Pistetään laite uuniin 1,5:ksi päiväksi.
7. Uunissapitoajan kuluttua otetaan laite uunista pois ja luetaan data muistipiiristä.
8. Verrataan saadut tiedot alkuperäiseen dataan.
9. Jos on uudet virheet, niiden osoitteet otetaan talteen.

5.8 Flash- ja FTL-algoritmin tilasto

Tämä ei ole varsinainen testitapaus, vaan yritys estimoida ja ennakoida valmiin tuotteen elinaikaa aikaisemmin saadun informaation perusteella. Flash- ja FTL-algoritmit firmwaren tasolla käyttävät samoja metodeja suorittamalla muistipiirin perusoperaatioita (luku-, kirjoitus- ja poisto-operaatioita). Täten voi luoda tarvittavaa informaatiota sisältävän yleisstruktuurin, johon liittyvät seuraavat parametrit: luku-, kirjoitus- ja poisto-operaatioiden laskurit sekä mapattujen virtuaalilohkojen lista. Tiedettäessä muistipiirin lohkojen lukumäärä luodaan taulu, jonka koko on muistilohkojen lukumäärä (1024). Taulun komponentit osoitetaan muistilohkon osoitteen perusteella. Tällä tavalla täytetään datalla tilastotaulu, joka on kiinnitetty fyysisten lohkojen osoitteisiin.

Firmwaren flash-operaation jälkeen lähetettiin signaali protolaitteeseen ja siten saatiin tilastodata protolaitteesta ulos. FTL-algoritmin tilastodata saatiin ulos trace-ohjelmiston avulla lähettämällä tietyn käskyn laitteen firmwarelle. Datan jälkikäsittelyoperaation suoritettua tiedot on sijoitettu MS

Excelin tiedostoon automaattisesti. MS Excel rakensi tarvittavat kaaviot saadun datan perusteella.

6 TESTITAPAUSTEN TULOSTEN KÄSITTELY

Testattavina oli neljä muistipiiriä: kaksi Micronin muistipiiriä ja kaksi Toshiban muistipiiriä. Tulosten näkökulmasta Micronin muistipiirit vaikuttivat Toshiban piirejä paremmilta. Toisaalta otetaan huomioon lopputuotetta koskevia asiakkaan vaatimuksia. Näitä ovat: datan käsittelynopeus, hinta ja muistipiirin fyysinen koko. Täten Toshiban tekemät muistipiirit sopivat Renesasin suunnittelemaan tuotteeseen ja täydellisesti tyydyttävät asiakkaan vaatimuksia kyseessä olevaa tuotetta kohtaan. Datan käsittelynopeuteen vaikuttaa ECC-algoritmin luonne. Testattavina olevissa Micronin muistipiireissä ECC-algoritmin toiminnallisuutta on ohjattu host-prosessorilla (ECC algoritmi on toteutettu SW-tasolla). Siten otetaan pois prosessoriaikaa, jota voitaisiin käyttää muiden operaatioiden suorittamiseen. Toshiban muistipiireissä ECC:n toiminnallisuutta hoidetaan muistipiirin mikrokontrollerilla säilyttämällä host-kontrollerin aikaa. Toshiban piireissä käytetään 8-bittinen ECC:tä, Micronin piireissä 4-bittinen ECC:tä. Toshiban valitun algoritmin tehokkuuteen vaikutti aikaisemmin mainittu muistipiirin fyysinen koko. Toshiban muistipiiri on 24 nm:n luokassa (29, s. 2).

Testauksen aikana saadut tulokset auttavat vastaamaan asiakkaan muistipiiriä koskeviin kysymyksiin. Testitapauksilla pyrittiin haastamaan muistipiirien dokumentoituja ominaisuuksia ja vertaamaan keskenään kahden eri tuottajan muistipiirejä.

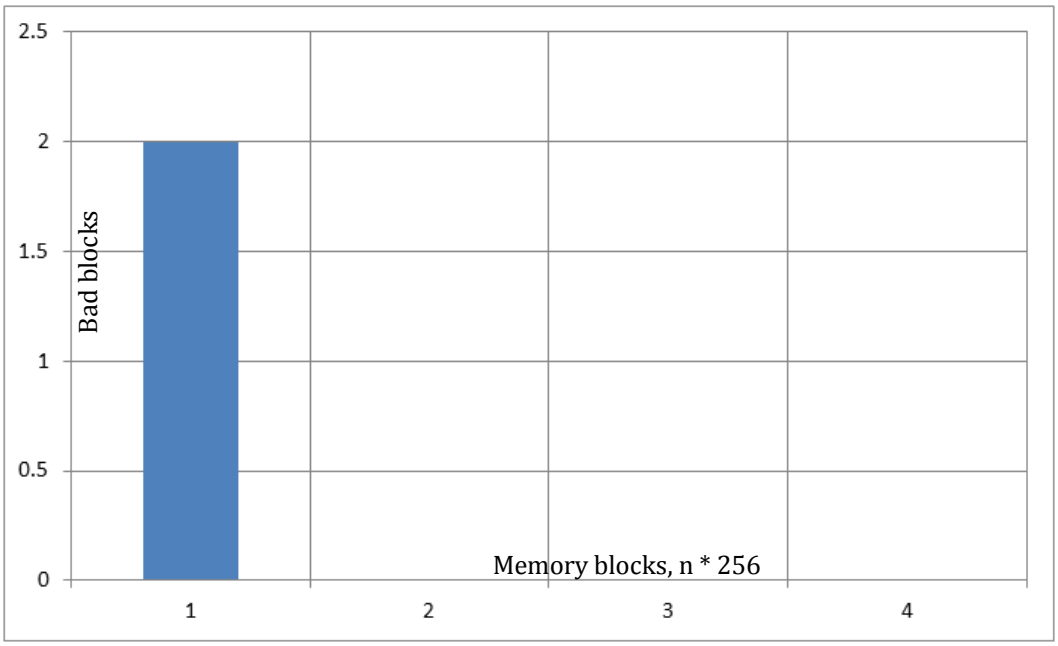
Aikaisemmin mainitun Toshiban built-in ECC-algoritmi ei vaikuttanut testitapausten tuloksiin. Muistipiirien valmistaja tarjoaa ECC Status Read -ominaisuuden, jonka avulla voi selvittää tapahtuvien virheiden lukumäärää lukuoperaation aikana. Sitä paitsi, voi selvittää paikan, jossa virhe tapahtui. (28, s. 38.)

6.1 Bad Blocks -testitapauksen tulokset

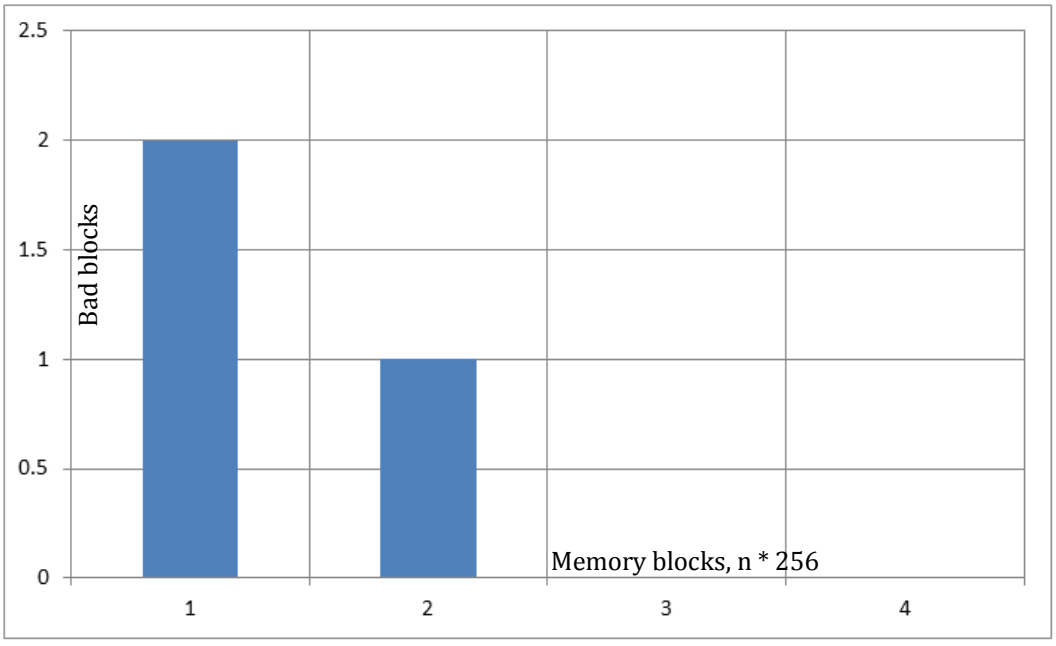
Bad Blocks -testitapauksen tavoitteena oli huonojen lohkojen alustavan kartan rakentaminen. Lisäksi testauksessa tarkastettiin järjestelmän tärkeimmän vaatimuksen täyttäminen, eli muistipiirien ensimmäisen muistilohkon on oltava aina hyvä.

Testatuista Micronin muistipiireistä ei löytynyt yhtään tehtaalta tullutta huonoa muistilohkoa. Toshiban ensimmäisistä piireistä löytyi kaksi huonoa lohkoa ja toisesta kolme. Täytyy myös ottaa huomioon tieto, mistä muistipiirissä olevista muistilohkoista huonot lohkot löytyivät. Tätä varten jaetaan muistipiirin hyödyllinen alue neljäksi osaksi. Jokaisessa osassa on 256 muistilohkoa.

Toshiban ensimmäisessä muistipiirissä lohkot 144 ja 207 on merkitty huonoiksi tehtaassa suoritetun testauksen jälkeen, toisessa muistipiirissä 97, 114 ja 274 (kuvat 28 ja 29).



KUVA 28. Toshiban ensimmäisessä muistipiirissä huonojen lohkojen lukumäärä per 256 lohkoa



KUVA 29. Toshiban toisessa muistipiirissä huonojen lohkojen lukumäärä per 256 lohkoa

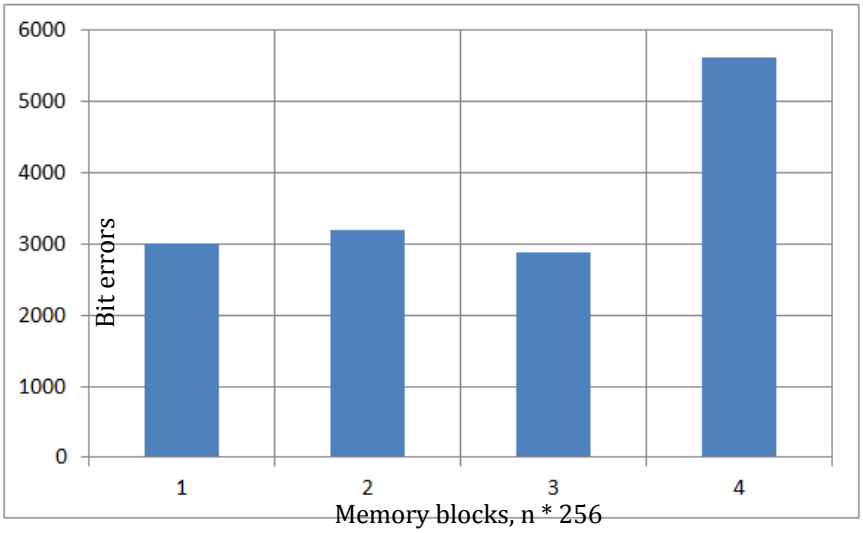
Kuvista 28 ja 29 voi todeta, että huonon muistilohkon esiintyminen 256 ensimmäisessä lohkossa on todennäköisempää. Täten on todennäköisempää, että Toshiban muistipiirissä ensimmäinen muistilohko on huono tehtaassa tehtyjen testien jälkeen.

Seuraava aspekti, joka täytyy myös ottaa huomioon, on huonojen lohkojen lukumäärä ja muistipiirin lohkojen lukumäärä. Huonojen lohkojen maksimilukumäärä on 20 ja NAND-ajurien implementoinnissa tämä on otettu huomioon. Jos huonojen lohkojen lukumäärä on suurempi kuin 20, siinä tapauksessa firmwaren lataaminen ei onnistu. Yllä mainitusta voi todeta, että Toshiban muistipiiriä käyttävään protolaitteeseen firmwaren lataaminen epäonnistuu todennäköisemmin kuin vastaava operaatio Micronin muistipiiriä käyttävässä laitteessa. Toshiban muistipiirissä huonojen lohkojen lukumäärä on suurempi kuin Micronin vastaavissa komponenteissa huonojen lohkojen lukumäärää tehtaassa tehtyjen testioperaatioiden jälkeen.

6.2 Data Retention -testitapauksen tulokset

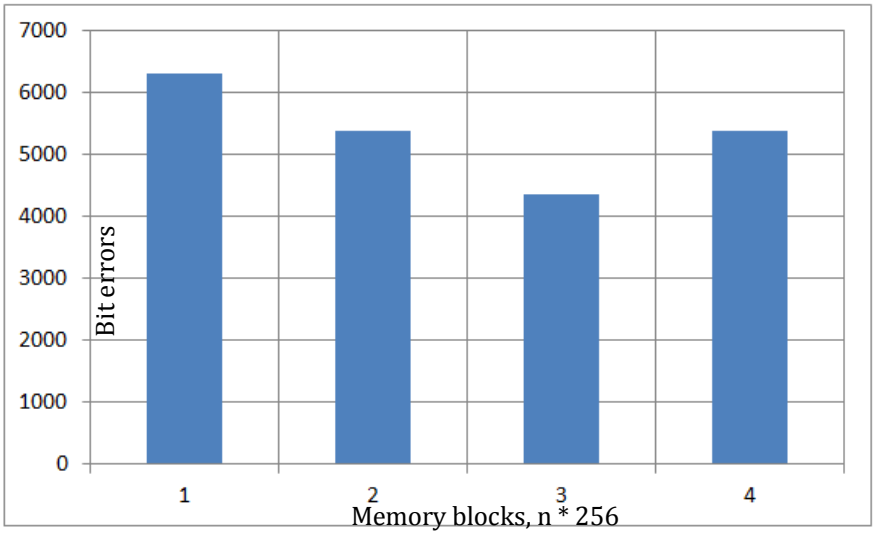
Data Retention -testitapauksen päätavoitteena oli tarkastaa muistipiirin kykyä säilyttää data muuttumattomana dokumentaatiossa mainitun aikakauden sisällä. Kumpikaan muistipiireistä ei päässyt testitapauksen läpi ilman virheitä. Täytyy kuitenkin mainita, että Micronin valmistamat muistipiirit vaikuttivat paremmilta kuin Toshiban tekemät piirit.

Micronin muistipiireissä ei tapahtunut yhtään virhettä, jota ei voisi korjata ECC-algoritmin avulla, sillä esiintyi vain 6–10 viallista bittiä per muistilohkon muistisivu. Virheellisten sivujen lukumäärä on suurempi muistipiirin 256 viimeisessä lohkossa kuin piirin muissa lohkoissa. Tilaston esityksessä koko muistipiiri on jaettu neljäksi osaksi. Kuvassa 30 on esitetty virheellisten sivujen lukumäärä per 256 lohkoa. Täytyy myös mainita, että kuvassa on esitetty virheellisten sivujen lukumäärien keskiarvot.



KUVA 30. Micronin muistipiirin virheellisten sivujen jakauma

Toshiban valmistamissa muistipiireissä viallisten bittien lukumäärä oli suurempi kuin Micronin muistipiireissä. Lisäksi datan lukuoperaation suoritettua on löydetty 5 % sivuja, joissa built-in ECC-algoritmi on epäonnistunut korjaamaan virheitä. Viallisten lohkojen lukumäärä on suurempi muistipiirin 256 ensimmäisessä lohkossa. Sivujen lukumäärä, joissa built-in ECC-algoritmi on epäonnistunut, on suurempi taas 256 ensimmäisessä sivussa, eli noin 3 % niiden kokonaismäärästä. Kuvassa 31 on esitetty virheellisten sivujen jakauma.



KUVA 31. Toshiban muistipiirin virheellisten sivujen jakauma

Kuvista 30 ja 31 voi todeta, että muistipiirin elinajan loppuvaiheessa Toshiban muistipiirin sisältävän protolaitteen käynnistäminen tai firmwaren päivittäminen epäonnistuu todennäköisemmin kuin vastaavan protolaitteen mainitut operaatiot, johon on rakennettu Micronin

muistipiiri. Toisin sanoen, on hyvin todennäköistä, että Toshiban muistipiirin ensimmäinen lohko on huono piirin elinajan loppuvaiheessa.

Tulosten perusteella myös voi todeta, että huonojen lohkojen lukumäärä kasvaa nopeammin Toshiban valmistamissa muistipiireissä niiden elinaikana.

6.3 Disturb- ja Partial Programming -testitapausten tulokset

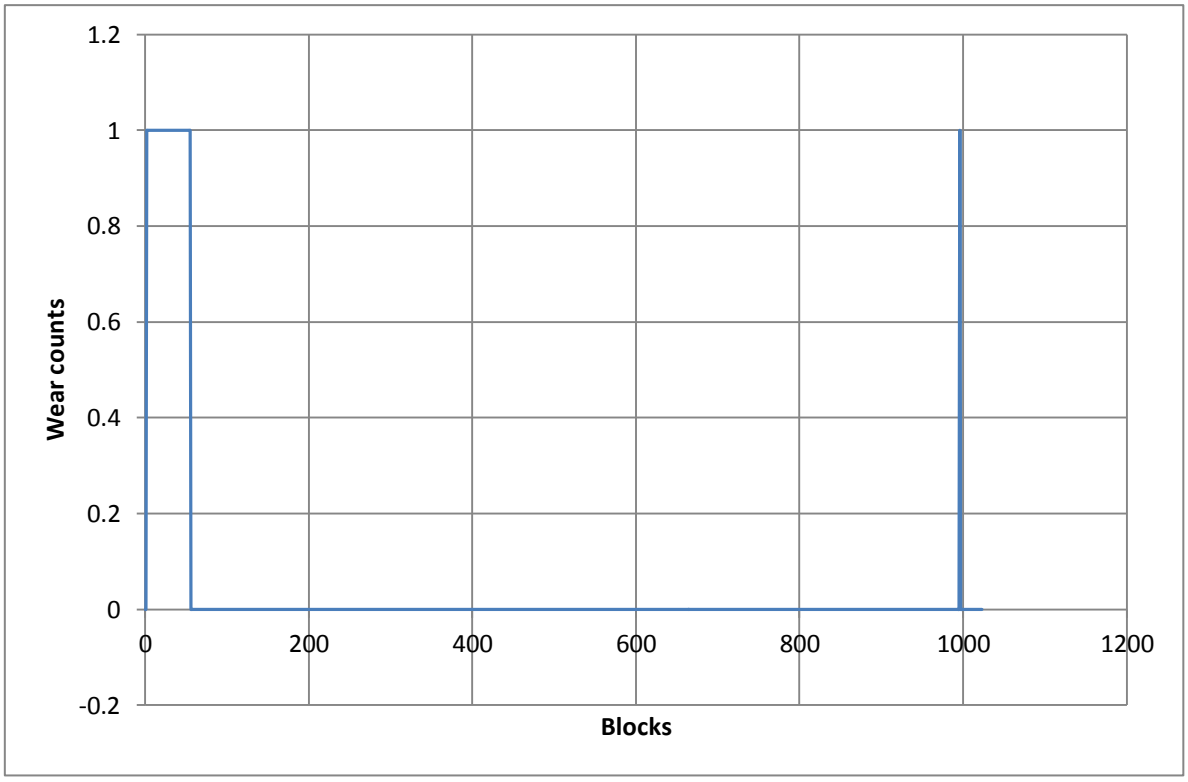
Disturb- ja Partial Programming -testaus on näyttänyt samanlaiset tulokset Toshiban ja Micronin muistipiireille. Disturb-testitapaukset eivät aiheuttaneet yhtään viallista bittiä.

Kaikki piirit pääsivät Partial Programming -testitapauksen läpi. Sitä paitsi on mainittava, että muistipiirien sivut kestävät keskimääräisesti 7 Partial Programming -operaatiota. Luvussa 5.6 mainittiin, että muistisivun on kestettävä neljää osittaista kirjoitusoperaatiota, jotta taataan järjestelmän oikea toiminta, koska jokainen sivu on jaettu neljäksi sektoriksi. Kuitenkin muistipiirien elinajan loppuvaiheessa osittaisten kirjoitusoperaatioiden lukumäärä vähenee. Tämä johtuu solujen oksidin heikentymisestä. Tässä vaiheessa voi ennakoida, että esille tulevat kaikki mahdolliset häiriöt, jotka aiheuttavat viallisia bittejä. ECC-algoritmi ei aina pysty korjaamaan mainittuja viallisia bittejä lukuoperaatioiden aikana. Tämä vuorostaan vaikuttaa laitteen järjestelmän toimintaan, eli laitteen firmwaren päivitys epäonnistuu tai laite ei käynnisty ollenkaan.

6.4 Program/Erase -testitapaus vs. Flash- ja FTL-operaatioiden suorittamisen tilastoa

Program/Erase -testitapauksen päätavoitteena oli haastaa muistipiirien dokumentaatiossa esitetyt kirjoitus- ja poistosyklien lukumäärät. Kumpikaan testauksen alla olevista muistipiireistä ei kestänyt 100 000:ta kirjoitus- ja poistosykliä. Kaikki testitapaukset päättyivät poisto-operaation epäonnistumiseen keskimääräisesti 90 000 syklin kuluttua. Täten muistipiirien elinaika on noin 10 % valmistajien lupaamaa aikaa lyhyempi. Tämä myös vaikuttaa protolaitteen järjestelmän elinaikaan negatiivisesti. Uusien huonojen lohkojen syntyminen tapahtuu nopeammin ja siten järjestelmän elinaika lyhenee.

Flash-operaation avulla saadun tilaston perusteella voi todeta, että firmware tallennetaan muistipiirin samaan muistialueeseen. Jos alueesta löytyy vanhaa SW, pyyhitään vanhat tiedot pois ja kirjoitetaan uudet samaan muistialueeseen. Tämä koskee ainakin 60 ensimmäistä muistilohkoa (kuva 32). Jos muistipiirissä on tarpeeksi vapaata tilaa (kuvat 33 ja 34), siinä tapauksessa kirjoitetaan informaatiota muualle, mutta jopa siinä tapauksessa 60 muistilohkon informaatio pyyhitään aina pois muistipiiristä.



KUVA 32. Firmwaren lataamisen operaatio edelleen käytettyyn laitteeseen (re-flash operation)

Tiedettäessä flash-operaatioon vaadittu aika, kirjoitus- ja poistosyklien maksimilukumäärä, kirjoitettavan informaation koko ja huonojen lohkojen maksimilukumäärä voidaan laskea laitteen maksimielinaika. Oletuksena on tieto, että suoritetaan vain flash-operaatiot. Kaavan 7 mukaisesti saadaan edelleen mainitun protolaitteen elinaika. (12, s. 1.)

$$Elinaika = \frac{Kirjoitus/Poisto\text{ -syklit} * käytössä\ olevien\ lohkojen\ lukumäärä}{käytettävien\ lohkojen\ lukumäärä * päivitykset\ yhden\ tunnin\ sisällä * tuntien\ lukumäärä\ vuorokaudessa} \quad (7)$$

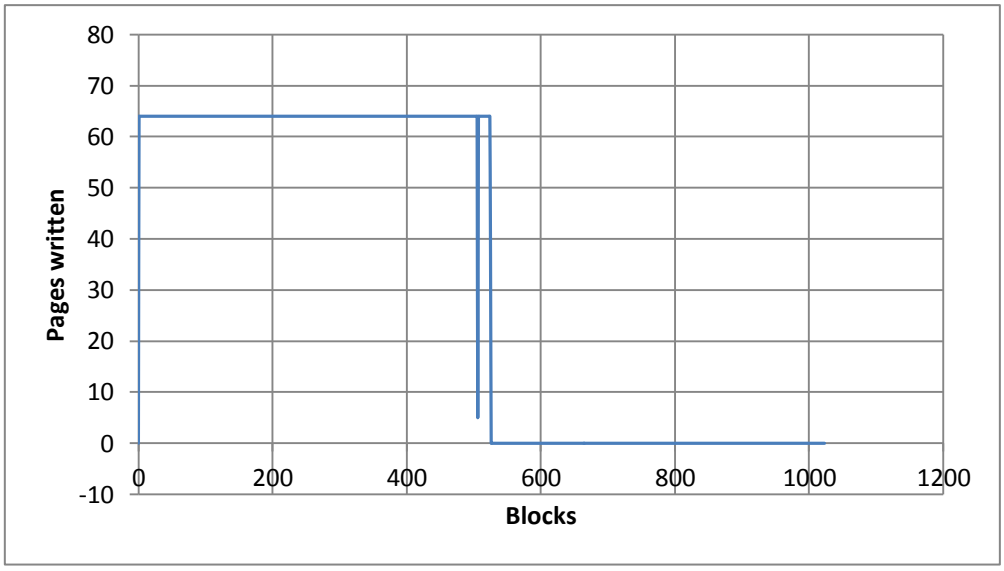
Kaavan 8 avulla lasketaan päivityksien lukumäärä tunnin sisällä.

$$Päivitykset\ tunnissa = \frac{sekuntien\ lukumäärä\ tunnissa}{päivityksen\ vaatima\ aika\ sekunneissa} \quad (8)$$

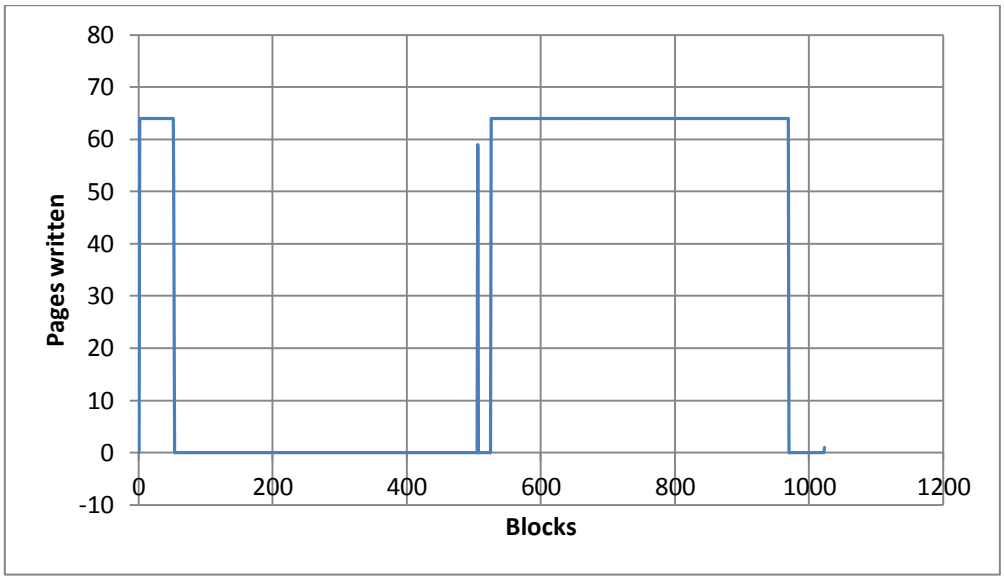
$$Päivitykset\ tunnissa = 3600 / 3 * 60 = 3600 / 180 = 20$$

$$Elinaika = \frac{90\,000 * 60}{60 * 20 * 24} = 187{,}5\ päivää$$

Kahden ensimmäisen flash-operaation jälkeen pyyhittävien muistilohkojen lukumäärä kasvaa noin 500 lohkoksi. Tämä voidaan päätellä kuvissa 33 ja 34 esitettyjen kirjoitusoperaatioissa käytettävien muistialueiden ja muistilohkojen lukumäärien perusteella.



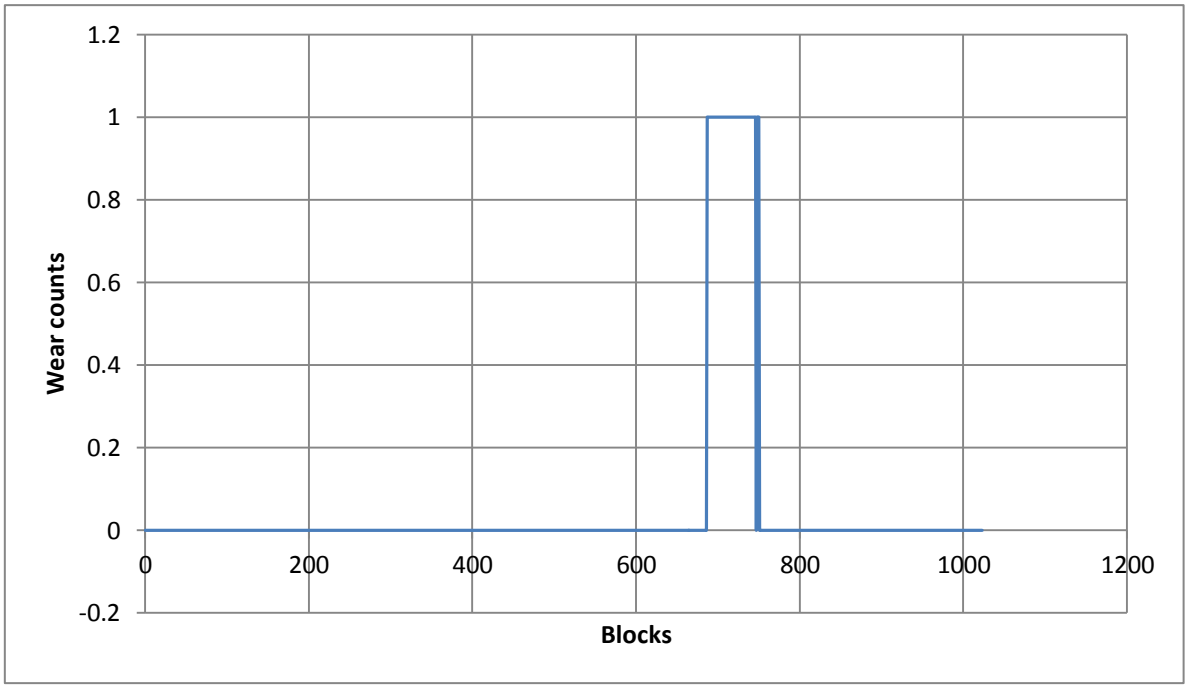
KUVA 33. Flash-operaation suorittaminen tyhjässä muistipiirissä



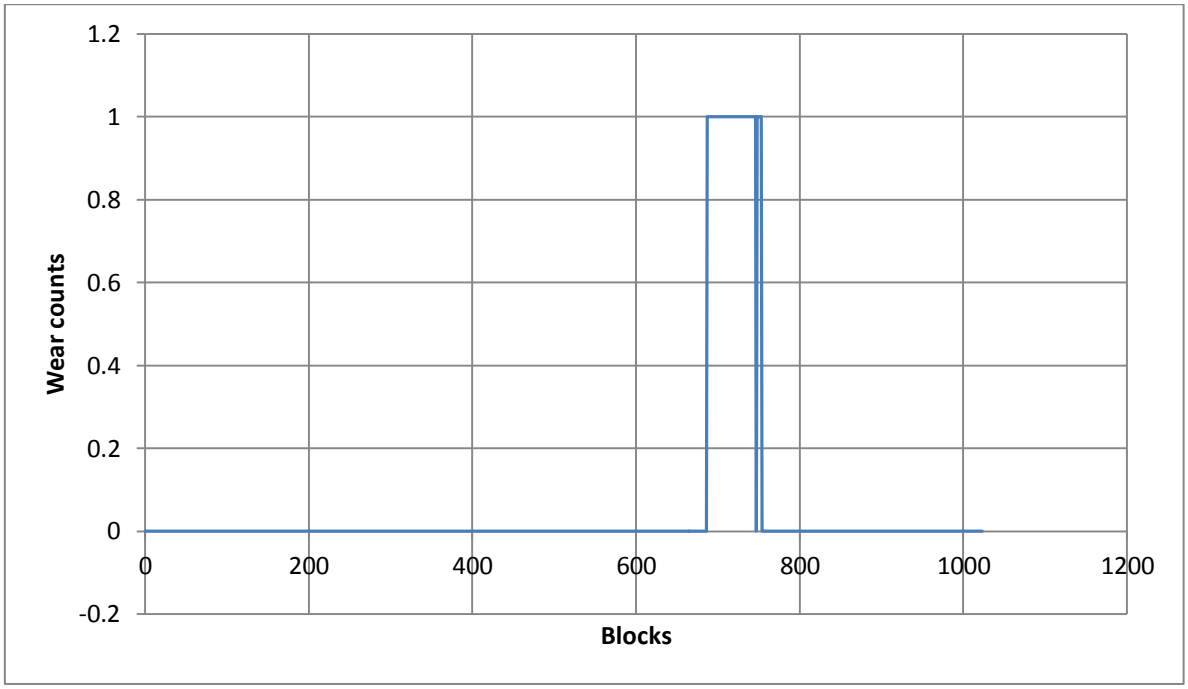
KUVA 34. Re-flash -operaatio

Vakituisesti pyyhittävien muistilohkojen lukumäärä (60) on huonojen muistilohkojen sallittua lukumäärää suurempi. Siten aiemmin saatu elinaika on myös protolaitteen järjestelmän elinaika (187,5 päivää). Laitteen toiminta mainitun skenaarion mukaisesti on hyvin mahdotonta ja sitä käytettiin vain näyttämään flash-operaation vaikutusta muistipiirin elinaikaan.

Täytyy kuitenkin mainita protolaitteen käytöksestä ajan mukaisesti. Kuvissa 35 ja 36 on esitetty wear countin tiedot heti laitteen käynnistyksen jälkeen ja kolmen tunnin päästä vastaavasti.



KUVA 35. Wear countin tiedot heti käynnistyksen jälkeen

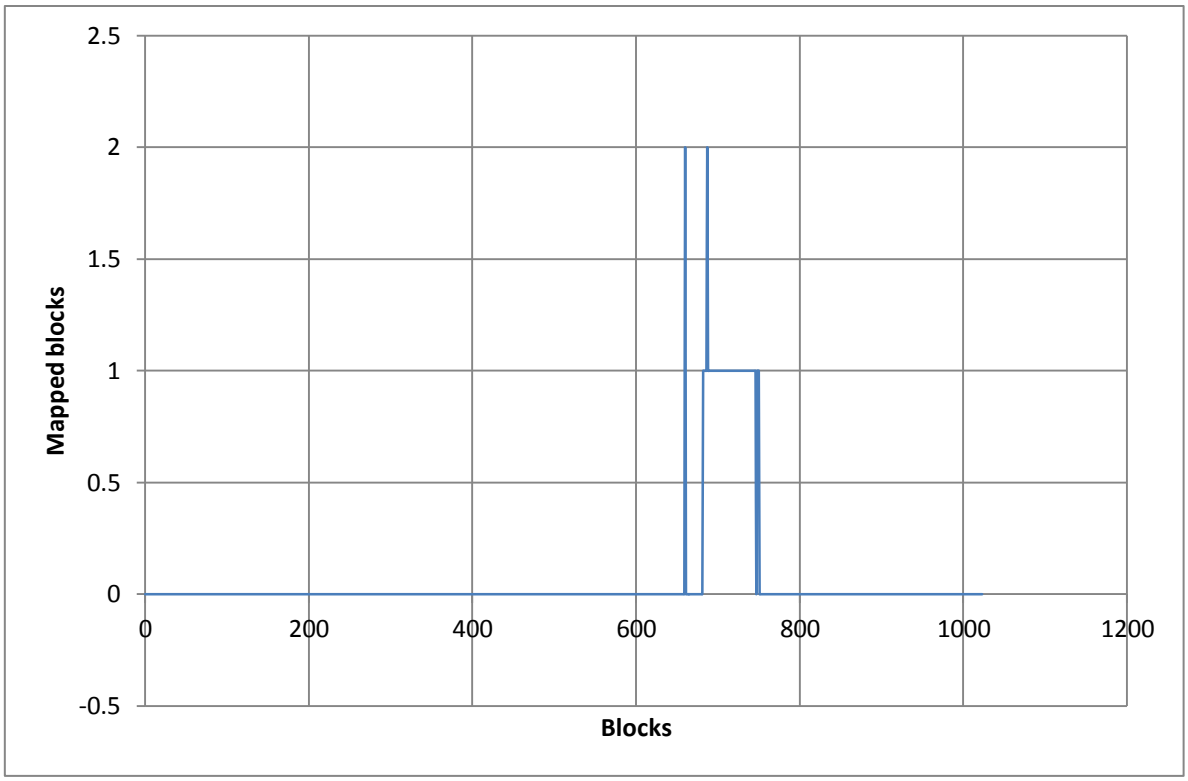


KUVA 36. Wear countin tiedot kolmen tunnin päästä

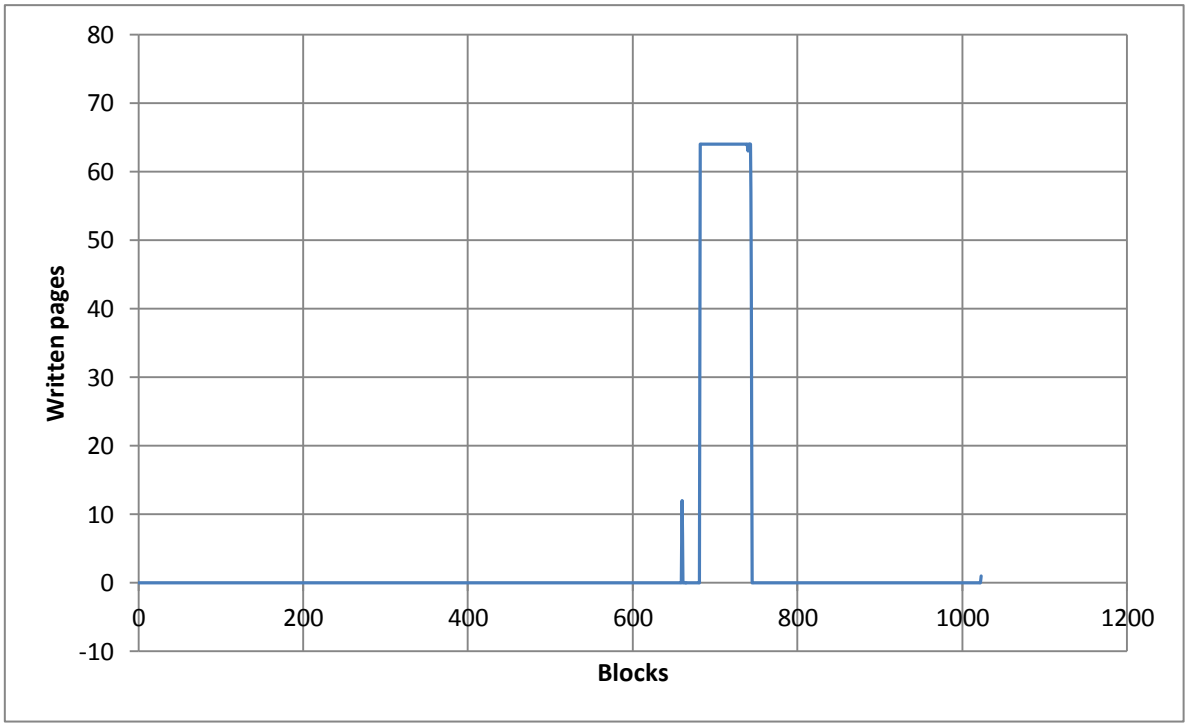
Kuten voidaan huomata kuvista 35 ja 36, protolaite ei käytä aktiivisesti muistipiiriä. Kolmen tunnin päästä vain seitsemän uutta sivua otettiin käyttöön. Täytyy myös mainita, että protolaite ei ollut kytketty tietoverkkoon, eli muistipiiriä käytettiin pakollisten tukioperaatioiden suorittamiseen. Muistioperaatioiden lukumäärä kasvaa kytkettäessä protolaitteen tietoverkkoon, eli kirjoitus- ja

poisto-operaatioiden lukumäärä sekä lukutoimintojen määrä. Kuvasta 36 huomataan myös, että poisto-operaatiota suoritetaan tavallisesti ennen uusien tietojen tallentamista muistipiirille.

Kuvista 37 ja 38 voidaan huomata, miten suoritetaan informaation kirjoittaminen virtuaalilohkoista fyysisiin muistilohkoihin. Mainituissa kuvissa on esitetty mapattujen lohkojen informaatio ja kirjoitettujen sivujen lukumäärät.



KUVA 37. Mapattujen sivujen informaatio heti käynnistyksen jälkeen



KUVA 38. Kirjoitettujen sivujen lukumäärä heti käynnistyksen jälkeen

Kuvasta 37 voidaan huomata, että kahdella fyysisellä lohkolla on ollut kaksi mapattua sivua ja sen alla olevassa kuviossa kirjoitettujen sivujen lukumäärät ovat vastaavasti 12 ja 64. Mainitusta tiedosta pääteltiin, että yhteen fyysiseen lohkoon voidaan hyvinkin kirjoittaa tietoja useammista virtuaalilohkoista ennen uuden lohkon käyttöönottamista. Siten säilytetään muistipiirin vapaita lohkoja ja estetään muistin fragmentaatiota. Kuvista 37 hyvin näkyy myös toinen asia, että kuitenkin monissa fyysisissä muistilohkoissa on käytetty kaikki 64 sivua ensimmäisen mappaamisen jälkeen. Täten käytettäessä laitetta aktiivisesti voidaan käynnistää kierrättämismekanismi aika pian. Edellä mainitusta johtuu, että sivujen wear count -parametri rupeaa kasvamaan suhteellisen nopeasti ja siten muistipiirin elinaika lyhenee. Sitä paitsi, sivujen wear count -parametri voi alkaa kasvaa kirjoitettaessa tietoja, kun muistipiirillä ei ole tarpeeksi vapaata tilaa.

Lukuoperaatioiden tilasto vain osoittaa sen, että lukeminen tapahtuu muita operaatioita useammin. Lisäksi tilastosta voidaan huomata, mitä alueita järjestelmä lukee muita useammin käynnistyksen aikana ja sen jälkeen. Sitä paitsi kannattaa ottaa huomioon se tieto, että käytettäessä muistipiiriä vain tukioperaatioiden suorittamiseen kuitenkin monet sivut luetaan yli 150 kertaa. Toisin sanoen, aktiivinen käyttö voi aiheuttaa sivujen usein toistuvaa lukuoperaatiota. Käytössä oleva FTL-algoritmi 1 000 000 lukuoperaation jälkeen päivittää sivun. Mainittu algoritmi

poistaa vanhat tiedot ja sen jälkeen tallentaa poistetut tiedot uudestaan kierrätyksen alla olevaan sivuun. Siten wear count -parametri kasvaa, mikä taas aiheuttaa muistipiirin elinajan lyhenemistä.

7 JOHTOPÄÄTÖKSET

Tutkimuksen päätavoitteena oli kahden eri muistipiirien valmistajan tuotteiden vertailu ja dokumentoitujen piirien ominaisuuksien ja rajoituksien haastaminen. Lisäksi sivutavoitteena tutkittiin flash-operaation suorittamisen ja FTL:n toiminnan vaikutus muistipiirin elinaikaan.

Testauksen tulosten näkökulmasta Micronin valmistama muistipiiri vaikuttaa paremmalta Toshiban tuottamaan muistipiiriin verrattuna. Micronin muistipiirit kestivät paremmin Data Retantion -testitapauksen, sillä piireistä ei löytynyt yhtään huonoa lohkoa tehtaassa tehdyn testauksen jälkeen. Sitä paitsi edelleen mainitun testin jälkeen enemmän virheitä löytyi muistipiirin 256 viimeisestä lohkosta, mitä vuorostaan on todella tärkeä protolaitteen järjestelmän toiminnan näkökulmasta, koska ensimmäisen lohkon on oltava aina hyvä ja virheetön. Löydetyissä virheissä ei ollut ollenkaan sellaisia virheitä, joita käytössä oleva ECC-algoritmi ei olisi onnistunut korjaamaan.

Toisaalta täytyy ottaa huomioon muistipiirin hintaa, kokoa ja datan käsittelynopeutta koskevia asiakkaan vaatimuksia. Mainitun mukaisesti Toshiban tuotteet kuitenkin alkavat vaikuttaa paremmilta, sillä muistipiirit ovat 24 nm:n luokasta sekä piirit ovat Micronin tuotteita suhteellisesti halvempia ja datan käsittelynopeus kolminkertaistuu verrattuna Microniin. Luvussa 6 on selitetty hyvin, miksi Toshiban tuottama muistipiiri on Micronin valmistamaa muistipiiriä nopeampi ja miksi Toshiban muistipiirin built-in ECC-algoritmi ei vaikuttanut testauksen tuloksiin.

Luvussa 6.4 tehdyn laskelman ja Program/Erase -testitapauksesta saadun datan perusteella voidaan todeta, että molempien valmistajien piirit kestävät hyvin aktiivista käyttöä. FTL-algoritmi hoitaa kaikkien tarvittavien rutiinien suorittamisen ja pidentää muistipiirin elinaikaa sekä estää mahdollisia häiriöitä ja virheitä datan käsittelyssä. Voi myös sanoa, että ongelmat esiintyvät datan käsittelyssä todennäköisemmin asiakkaan valitseman muistipiirin elinajan loppuvaiheessa kuin Micronin tekemän tuotteen samassa elinajan vaiheessa. Sitä paitsi Toshiban muistipiirin sisältävän laitteen joko käynnistäminen tai firmwaren päivittäminen epäonnistuu todennäköisemmin kuin samat operaatiot laitteessa, jossa on Micronin valmistama komponentti.

8 POHDINTA

Työn päätavoitteena oli verrata kahden eri valmistajan piirejä keskenään ja haastaa tuottajien dokumentoimia muistipiirien parametreja. Tutkimuksen sivutavoitteena oli arvioida flash-operaatioiden ja FTL:n toiminnallisuuden vaikutus muistipiirin elinaikaan. Tavoitteiden saavuttamiseksi suoritettiin teorian mukaisesti suunniteltuja ja toteutettuja testitapauksia. Tarvittavan informaation etsintä ja protolaitteistoon tutustuminen ovat vaatineet eniten aikaa työn kokonaisajasta, vaikka informaation kerääminen oli aloitettu ennen työsopimuksessa määritettyä töiden alkupäivää. Testitapauksien toteuttaminen verrattuna teoreettiseen selvitykseen oli tehdyn tutkimuksen helpoin osa. Vaikka kaikki oli suunniteltu hyvin ja paljon etukäteen, useasti piti palata takaisin ja harkita asiat uudelleen, joskus muuttaa testitapausten algoritmia ja suorittaa testaus uudestaan.

Tutkimuksen tavoitteet saavutettiin kaikesta huolimatta ja kattavasti. Tulosten perusteella voidaan sanoa, että asiakkaan valinta ei ole paras mahdollinen laadun ja laitteen järjestelmän toiminnan näkökulmista. Lopputuotteen päivittäminen tai käynnistäminen voi aiheuttaa ongelmia, joille ei ole SW-ratkaisuja, vaan on vaihdettava muistipiiri. Toisaalta valittu Toshiban muistipiiri mahdollistaa hyvin korkean datan käsittelynopeuden ja lopputuotteen pienen koon, mikä on vuorollaan todella tärkeä nykyaikaisessa elektroniikassa.

Jatkotutkimuksena esimerkiksi voisi tarkastaa lähtöjännitteen vaikutusta muistipiirien toimintaan. Toisaalta testauksen jatkokehitysideana voi selvittää hyvin korkean ja matalan lämpötilan vaikutusta NAND:in toiminnallisuuteen.

LÄHTEET

1. Overview of HW platforms and Mini Host SW architecture (strictly confidential). 2012. Renesas Mobile Corporation.
2. WGM-2.9-23 Power Management & System Control HW IF Spec WGModem2.9 Documentation (highly confidential). 2010. Renesas Mobile Corporation.

3. Mini Host Domain Software Architecture (highly confidential). 2012. Renesas Mobile Corporatio.

4. Top Level Platform Architecture NSYS-DES-0000020 (highly confidential). 2010. Renesas Mobile Corporation.

5. μiTron 4.0 specification (Ver. 4.00.00). 2002. Japan. ITRON Committee – TRON Association.

6. NAND flash memory. Saatavissa: http://whatis.techtarget.com/definition/NAND-flash-memory. Hakupäivä 29.6.2013.

7. Micheloni, Rino – Crippa, Luca – Marelli, Alessia. 2010. Inside NAND flash memory. New York, USA – Great Brittan, London: Springer Dordrecht Heidelberg.

8. Flash memory 101: An Introduction to NAND flash. 2006. EE Times. Saatavissa: http://www.eetimes.com/document.asp?doc_id=1272118. Hakupäivä: 29.6.2013.

9. Fisher, Ryan. 2008. Optimizing NAND Flash Performance. Micron Technology, Inc.

10. NAND Flash FAQ. Eureka Technology. Saatavissa: http://www.actel.com/ipdocs/apn5_87a_FAQ.pdf. Hakupäivä: 30.6.2013.

11. Introduction to NAND Flash Memory. Digi-Key Corporation. Saatavissa: http://dkc1.digikey.com/us/en/tod/Micron/NANDFlash_NoAudio/NANDFlash_NoAudio.html. Hakupäivä: 30.6.2013.

12. Jedrak, Michal – Evatronix S.A. NAND Flash memory in embedded systems. Saatavissa: http://www.design-reuse.com/articles/24503/nand-flash-memory-embedded-systems.html. Hakupäivä: 30.6.2013.

13. Windbacher, Thomas. 2010. Engineering Gate Stacks for Field-Effect Transistors. 2.1.1 Flash Memory. Saatavissa: http://www.iue.tuwien.ac.at/phd/windbacher/node14.html. Hakupäivä: 1.7.2013.

14. Open NAND Flash Interface Specification. 2012. Intel Corporation – Micron Technology, Inc. – Phison Electronics Corp. – SanDisk Corporation - SK Hynix, Inc. – Sony Corporation – Spansion. Revision 3.1.09.19.2012.
15. SLC vs. MLC: An Analysis of Flash Memory. Super Talent Technology, Inc. Saatavissa: http://www.supertalent.com/datasheets/SLC_vs_MLC%20whitepaper.pdf. Hakupäivä: 1.7.2013.

16. CTAN010: SLC vs. MLC NAND. 2008. Cactus Technologies Application Note. Saatavissa: http://www.systronics.ch/files/Application_Note_SLC_vs_MLC.pdf. Hakupäivä: 2.7.2013.

17. Tanzawa, Toru – Tanaka, Tmoharu – Takeuchi, Ken – Shirota, Riichiro – Aritome, Seiichi – Watanabe, Hiroshi – Hemink, Gertjan – Shimizu, Kazuhiro – Sato, Shinji – Takeuchi, Yuji – Ohuchi, Kazunori. A Compact On-Chip ECC for Low Cost Flash Memories. Saatavissa: http://www.takeuchi-lab.org/image/PublicationJ6.pdf. Hakupäivä: 2.7.2013.

18. Sylvester, Joel. 2001. Reed-Solomon Codes. Saatavissa: http://www.csupomona.edu/~jskang/files/rs1.pdf. Hakupäivä: 2.7.2013.

19. What Types of ECC Should Be Used on Flash Memory? 2011. Spansion. Saatavissa: http://www.spansion.com/Support/Application%20Notes/Types_of_ECC_Used_on_Flash_AN.pdf. Hakupäivä: 2.7.2013.

20. TN-29-61: Wear Leveling in NAND Flash Memory Introduction. 2011. Micron.

21. NAND Flash Memory MT29F1G08ABAEAWP-IT, MT29F1G08ABAEAWP, MT29F1G08ABAEAH4-IT, MT29F1G08ABAEAH4, MT29F1G08ABBEAH4-IT, MT29F1G08ABBEAH4, MT29F1G16ABBEAH4-IT, MT29F1G16ABBEAH4, MT29F1G08ABBEAHC-IT, MT29F1G08ABBEAHC, MT29F1G16ABBEAHC-IT, MT29F1G16ABBEAHC. 2011. Micron. Datalehti.

22. TC58DYG02D5BAI6 datasheet. 2012. Toshiba.

23. TN-29-59: Bad Block Management in NAND Flash Memory. 2011. Micron.

24. TN-29-17: NAND Flash Design and Use Considerations. 2006. Micron.

25. Inoue, Atsushi – Wong, Doug. 2003. NAND Flash Applications Design Guide.

26. Venkat, Kripasagar – Haensel, Uwe. 2008. Application Report SLAA392.

27. Activation Energies of Failure Mechanisms in Advanced NAND Flash Cells for Different Generations and Cycling. 2013. IEEE Electron Devices Society. Saatavissa: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6461401&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F16%2F6466448%2F06461401.pdf%3Farnumber%3D6461401. Hakupäivä: 5.7.2013.

28. TC58BYG1S3HBAI6. 2013. Toshiba.

29. How to handle the increasing ECC requirements of the latest NAND Flash memories in your Industrial Design. Toshiba Electronics Europe.