UNIVERSITÀ DEGLI STUDI DI PALERMO

Dottorato in Energia e Tecnologie dell'Innovazione

Curriculum: Tecnologie dell'Informazione e Scienze Applicate

Dipartimento di Energia , Ingegneria dell'Informazione e Modelli Matematici

Settore Scientifico Disciplinare - ING/INF-01

TECNICHE DI TEST INNOVATIVE PER LA CARATTERIZZAZIONE DI MEMORIE A GATE FLOTTANTE

IL DOTTORE

ING. GINEUVE ALIERI

IL COORDINATORE

PROF.ssa MARIA STELLA MONGIOVI'

IL TUTOR

PROF. G. COSTANTINO GIACONIA

CO TUTOR

ING. FRANCESCO LA ROSA

CICLO XXIX

ANNO CONSEGUIMENTO TITOLO – 2017

Alla Grinta,

alla Determinazione,

alla Forza di non mollare Mai.

A Te, anzi a Voi,

che fino all'ultimo istante

mi avete insegnato tutto questo

e molto di più.

Grazie Mamma…

Grazie Nonna…

Ringraziamenti

Questi tre/quattro anni di Dottorato sono stati fra i più lunghi e difficili della mia Vita (che già di suo non è che fosse molto lineare), per fortuna ho avuto vicino delle persone che hanno creduto in me, mi hanno supportata, spronata, a volte rimproverata e grazie alle quali il mio carattere combattente è venuto fuori ancora una volta.

Vorrei iniziare col ringraziare il Prof. Ing. Giuseppe Costantino Giaconia, mio tutor universitario, per avermi guidato nel percorso di ricerca, supportandomi e spronandomi, sempre credendo nel mio operato.

Un grazie particolare va all'Ing. Leonardo Mistretta, mio collega nel laboratorio ESPDLAB dell'Università di Palermo, per la sua immensa disponibilità e il continuo feed-back ricevuto per la progettazione della scheda.

Grazie a tutto l'STMicroelectronics Development Center di Palermo che mi ha accolta per gran parte del mio percorso fornendomi tutti gli strumenti e il supporto necessari allo sviluppo del progetto. In particolare vorrei ringraziare l'Ing. Francesco La Rosa, mio tutor aziendale, l'Ing. Angelo Cimino, l'Ing. Ignazio Mocciaro e l'Ing. Salvatore Imbesi che mi hanno assistito dal punto di vista tecnico e professionale con grande competenza. Un grazie sincero e profondo va ad Angelo che, oltre a seguirmi tecnicamente, mi ha supportato enormemente da un punto di vista morale e personale, ricorda se decidi di cambiare mestiere puoi fare lo psicologo!

Un sentito ringraziamento al mio amico e compagno di corso, l'Ing. Fulvio Caruso.

Un Grazie, la cui grandezza non è misurabile, alla mia compagna di Vita, Antonella, senza La quale in molte occasioni avrei mollato, tu sai di cosa parlo…

Grazie a mia Sorella, Kerima, che in questi anni ho scoperto più grande di quanto immaginassi.

Grazie a Giorgio, Viola e Naswa (l'ordine è casuale :D*), che sono i miei fratelli/sorelle per scelta, e vi assicuro che questa scelta non è sempre facile dato il mio carattere per così dire…Particolare!*

Grazie a mio zio Vittorio, che spesso rivedo più come Papà che come zio!

Infine i ringraziamenti più difficili e semplici allo stesso tempo…

Grazie Nonna Agata! Tu sei sempre stata, in assoluto, la mia prima sostenitrice, hai sempre creduto in me, mi hai permesso di crescere e di studiare, questo ennesimo traguardo appartiene più a Te che a me…

Grazie Mamma, Tu sei sempre stata il mio modello di Donna! Tu…forte e determinata, sempre pronta a riconoscere i tuoi sbagli, a cadere e rialzarti, dolce ma riservata come poche, tu non eri per tutti ma c'eri per tutti…potrei continuare all'infinito, ma niente riuscirebbe a descrivere cos'eri! Spero che davvero, come dicono in molti, un po' ti assomiglio; spero di riuscire a portare avanti sempre tutto con la tua stessa forza; spero che Tu, Voi, siate fiere di me…

E Spero che, ovunque Voi siate, stiate bevendo alla facciazza mia!!!

Riempite il bicchiere alla Nonna!!!

Vi Amo…

Indice

Abstract

L'affidabilità nella ritenzione dei dati memorizzati è una delle problematiche fondamentali delle memorie flash; esse vengono normalmente testate, in produzione con procedure specifiche implementate su ATE (Automated Test Equipments), per rilevare problemi di lettura, programmazione e cancellazione; vengono inoltre provate altre procedure per identificare possibili faults e per il corretto trimming dei parametri interni.

Oggi, il testing classico con ATE è supportato dalle tecniche BIST (Built-In-Self-Test), tramite le quali si prevede in progetto, all'interno dei circuiti integrati, una parte di hardware e software supplementari per permettere l'auto-test (funzionale e/o parametrico), così da ridurre la dipendenza da apparecchiature ATE esterne e quindi riducendo sensibilmente il tempo di esecuzione del test completo e permettendo la realizzazione di test del circuito integrato in qualsiasi istante e condizione di funzionamento (in-field test).

Un ulteriore passo avanti nel testing in-field è stato effettuato con lo sviluppo delle tecniche SBST (Software-Based Self Test), che si basano sull'uso del microprocessore interno alla memoria per effettuare i test necessari via software, senza avere così la necessità di progettare e realizzare nel chip hardware aggiuntivo, così come accade nel caso di presenza di strutture BIST all'interno del circuito.

Rispetto allo stato dell'arte delle tecniche di testing si è approfondito lo studio, la progettazione e la realizzazione di una soluzione circuitale innovativa denominata Portable-ATE per il testing dei Memory Test Chip.

La scheda Portable-ATE è stata sviluppata partendo da una demo-board STM32-Nucleo (STM32F072RB) ed aggiungendo una scheda custom che permette di gestire il sistema di alimentazione, la corretta comunicazione con il memory test chip e l'alloggiamento del test chip stesso. Essa è in grado di valutare le perfomances di un singolo memory test-chip mantenendo gli stessi standard di affidabilità e riproducibilità di un ATE classico, aggiungendo portabilità, flessibilità, configurabilità e la possibilità di sviluppo e debug di algoritmi di gestione e software-based self test (SBST) con logging in tempo reale.

Con l'utilizzo del Portable-ATE è stata effettuata la caratterizzazione di alcuni memory test chip di STMicroelectronics con particolare attenzione alle analisi di stress, quale gate e drain stress e successivamente sono stati effettuati test di ciclatura per verificare la ritenzione dei dati; per valutare i test effettuati sono state estratte le distribuzioni di soglia.

Infine, grazie all'utilizzo del Portable ATE è stato possibile sviluppare, testare e ottimizzare gli algoritmi di gestione della memoria, program con verify, erase con verify e refresh senza avere all'interno della memoria nessun PEC (Program/Erase Controller) o microprocessore interno per la gestione degli stessi e con la possibilità, totalmente innovativa, di avere un logging in tempo reale delle operazioni in esecuzione.

1 Introduzione

Le memorie flash sono uno dei pochi esempi tecnologici il cui debutto fu un fallimento, infatti nel 1985 presso Toshiba è stata annunciata la tanto attesa cella di memoria a singolo transistor cancellabile elettricamente, inventata dall'Ing. Masuoka [1]. Questa nuova tecnologia fu battezzata "Flash Memory", data la sua capacità di essere cancellata in un flash, ma il suo lancio fu fallimentare date le sue iniziali difficoltà di utilizzo.

Successivamente, nello stesso anno, anche Intel sviluppò all'interno dei suoi laboratori la prima flash Memory a 64Kb, ma dato il precedente fallimento di Toshiba non volle commercializzare il prodotto. Solo l'insistenza di Gordon Moore e del suo team portarono nel 1986 alla commercializzazione da parte di Intel di una Flash Memory a 256Kb.

Grazie alla quasi totale compatibilità del processo produttivo (≈95%) fra EPROM e Flash la nuova tecnologia riuscì velocemente ad ottenere gli stessi volumi di produzione delle EPROM e successivamente a superarli enormemente.

Nel giro di pochi anni le memorie flash hanno subito una notevole evoluzione quale ad esempio la divisione della memoria in blocchi, indipendentemente cancellabili, l'integrazione all'interno del chip tutti i circuiti di controllo necessari alla programmazione e cancellazione in maniera automatica, l'integrazione delle pompe di carica così da non richiedere l'impiego di differenti valori di alimentazione, alcuni dei quali di elevato valore rispetto agli standard digitali necessari fino a quel momento per il corretto funzionamento.

Queste evoluzioni hanno permesso una veloce scalata del mercato delle memorie, che hanno seguito le performances della così detta la legge di Moore, mostrata in Figura 1; inoltre si è passati da un costo di 1000$/MB del 1988 a circa 2cents/MB nel 2010. Oggi il mercato delle memorie flash è uno dei mercati principali nelle moderne industrie dei semiconduttori, dato il massivo utilizzo di questi dispositivi sia in modalità stand alone sia di tipo embedded, all'interno dei più comuni apparecchi elettronici.

Le stringenti specifiche richieste dai moderni dispositivi mettono in evidenza l'importanza di sviluppare memorie flash con prestazioni e affidabilità sempre migliori.

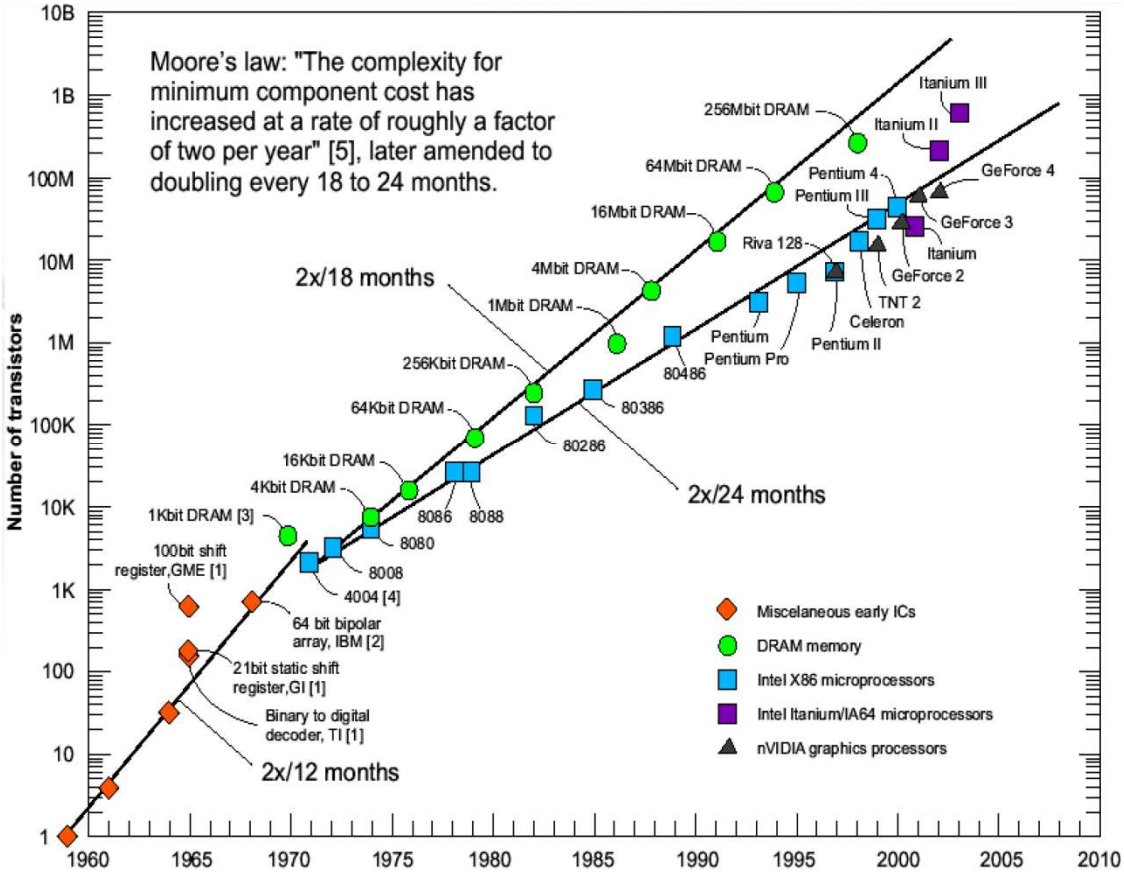


Fig. 1: Evoluzione delle tecnologie e delle grandezze delle memorie e dei microprocessori.

La continua diminuzione della dimensione delle celle elementari ed il conseguente aumento della capacità totale degli array e dei circuiti per la gestione della memoria stessa ha portato alla necessità di sviluppare modalità di test sempre più accurate, ad una sempre più elevata copertura dei guasti (fault coverage) e la contemporanea riduzione del costo stesso dei test. Infatti, il costo della fase di test di una memoria flash (ed in generale dei circuiti integrati) incide notevolmente sul costo globale di un chip.

In questo contesto si è sviluppato il mio progetto di dottorato, cercando nuove prospettive per il test delle memorie flash, tale da anticipare il più possibile la scoperta di eventuali bug e determinare delle tecniche di "early test" capaci di esplorare il comportamento di una memoria sin dalla fase di progettazione della stessa, onde poterne correggere gli eventuali errori in fase di ricerca e sviluppo (R&D – Research and Development).

Nella prima parte della tesi verranno illustrate le due principali tipologie di array delle memorie flash, NOR e NAND, con i relativi stress subiti dalle stesse a seguito del loro utilizzo: verranno pertanto illustrati i meccanismi principali di programmazione, cancellazione e lettura.

Analizzate le principali cause di stress degli array saranno mostrate le principali tecniche di test normalmente utilizzate, a partire dal test in produzione con Automated Test Equipment (ATE), per poi passare alle tecniche Built In Self Test (BIST) e ai transparent-BIST, per passare poi alle innovative tecniche Software Based Self Test utilizzate nei moderni microprocessori e negli ultimi anni introdotte per il testing delle memorie embedded.

Durante tale periodo si è anche sviluppata una scheda, denominata Portable-ATE, per il testing di un singolo memory test chip, il cui hardware e software sarà descritto nella seconda parte di questa digressione.

Con tale dispositivo è stata possibile la caratterizzazione e lo studio precoce di una nuova tipologia di memoria progettata da STMicroelectronics.

L'innovazione introdotta dal Portable ATE consiste nella possibilità una completa fase di test della memoria senza l'utilizzo di ATE classici ed in particolare sono stati condotte esaustive campagne di test relative ad ogni aspetto del memory test chip fra cui prove di drain stress, gate stress e test di ciclatura, come sarà mostrato nel capitolo 4. Inoltre, è stato possibile lo sviluppo ed il test diretto in memoria degli algoritmi di gestione quali program con verify, erase con verify e refresh, pur non essendo presente all'interno del memory test chip il microprocessore o il Program-Erase-Controller (PEC) completo. Si è inoltre effettuata l'analisi esaustiva dell'impatto di diverse tipologie di algoritmi, adoperate per effettuare le operazioni di erase con verify, sul tempo di vita della memoria.

Si è infine verificato, su alcuni test standard, che i risultati prodotti dal portable ATE fossero in linea con quelli prodotti dagli ATE tradizionale, onde validare il comportamento della nuova scheda, che peraltro ha la non secondaria caratteristica di essere un prodotto a basso costo, alta velocità e semplicità d'uso.

Come tale essa può sicuramente costituire un validissimo strumento di misura, una sorta di indispensabile "utensile" che arricchisce la cassetta degli attrezzi del progettista di memorie.

2 Flash Memory – Stress e Testing

In questo capitolo sarà esposta una breve panoramica sulle memorie flash, concentrando particolarmente l'attenzione sugli stress subiti dalle celle della memoria durante il suo funzionamento ed i relativi test che si utilizzano per rilevare eventuali malfunzionamenti.

Nei paragrafi seguenti si analizzerà il funzionamento e la struttura di un array flash, esaminando le operazioni svolte sulla memoria e gli eventuali disturbi causati dalle operazioni stesse. Successivamente si descriveranno le varie tipologie di test effettuate sulle memorie flash, a partire dai tipici test su ATE effettuati a fine produzione, passando per le tecniche di tipo Built-In-Self-Test (BIST) che utilizzano hardware dedicato al test ed implementato internamente dalla memoria alfine di poter rivelare ed in alcuni casi per autocorreggere gli eventuali fallimenti nell'esecuzione delle operazioni attuabili. Si descriveranno infine le innovative tecniche di tipo Software-Based-Self-Test (SBST) basate su software e che quindi non necessitano di hardware aggiuntivo integrato nel chip.

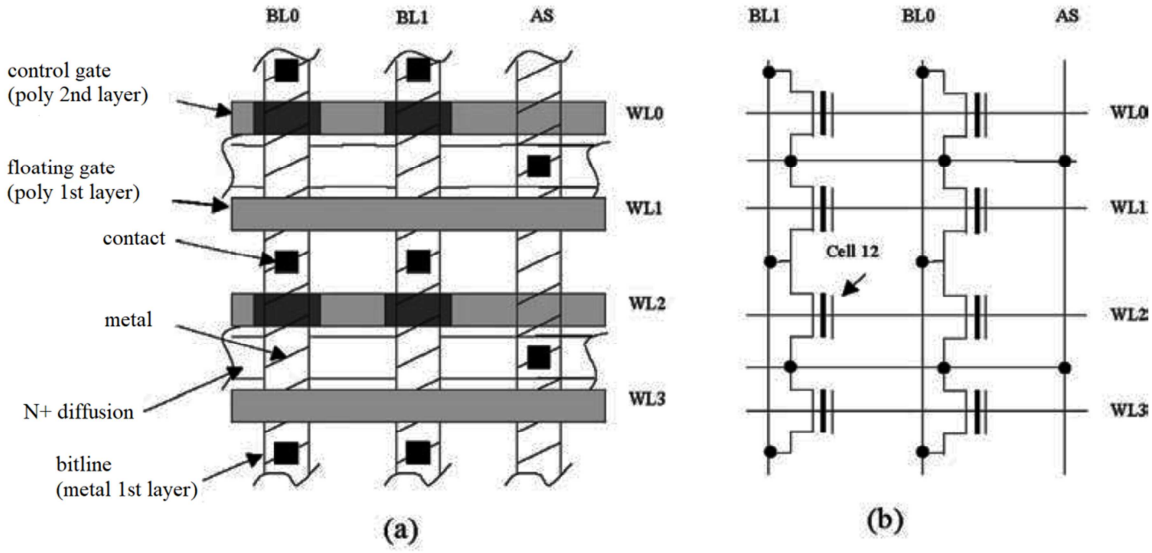
2.1 Tipologie di Array e relativi Stress.

Negli ultimi anni le memorie flash sono diventate la principale e più economica tecnologia di storage a stato solido e sono ampiamente utilizzate nei dispositivi elettronici mobili e nella quasi totalità di applicazioni consumer. In base alla loro architettura interna, queste si possono dividere in due principali tipologie: le Flash Memory ad architetture di tipo NOR e quelle di tipo NAND. All'interno del mercato odierno le memorie i principali attori delle architetture di tipo NOR sono Intel ed STMicroelectronics, mentre Toshiba e Samsung hanno nelle NAND Flash la loro feta di mercato predominante.

L'architettura di tipo NOR [1] è mostrata nella figura 2, dove è visibile lo schema prinicpale dei collegamenti, il relativo layout su fetta e le tipiche tensioni di funzionamento applicate nelle procedure di programmazione, lettura e cancellazione. In questa tipologia di memoria la programmazione avviene tipicamente per hot-electron injection [2], mentre la cancellazione avviene per tunneling di tipo Fowler-Nordheim [2].

Nelle memorie NOR le singole celle di memoria sono connesse in parallelo, in questo modo ogni cella può essere indirizzata singolarmente per scrittura o programmazione, garantendo in tal modo bassi tempi di accesso alla memoria in ipotesi di voler indirizzare la memoria con valore a piacimento (random access).

Questa tipologia di memoria risulta molto affidabile per lo storage dei dati, particolarmente indicata per applicazioni in cui necessitano alte velocità di lettura e densità di celle relativamente basse. L'uso più comune infatti è la memorizzazione storage di codice o di dati in applicazioni embedded e nei dispositivi elettronici portatili, quali i telefoni cellulari.



	BL0	BL1	AS	WL0	WL1	WL2	WL3
Programming Cell 12	Float	4-6V	0V	0V	0V	9-12V	0V
Reading Cell 12	0V	1V	0V	0V	0V	4-5V	0V
Array Erase	Float	Float	4-6V	-10V	-10V	-10V	-10V

(c)

Fig. 2 Flash Memory di tipo NOR: a) topologia del layout di un array, b) schematico equivalente, c) tensioni tipiche di funzionamento. (Figura 3.4 di [4])

Le flash NAND [3] sono state sviluppate come un'alternativa ottimizzata per l'archiviazione di dati ad alta densità, penalizzando il tempo di accesso alla memoria per ottenere una minore dimensione delle celle. Questo si traduce in una minore dimensione del chip e dunque nel minor costo per bit. Per ottenere ciò, a differenza delle memorie NOR, le celle di memoria vengono collegate in serie, come mostrato in figura 3.

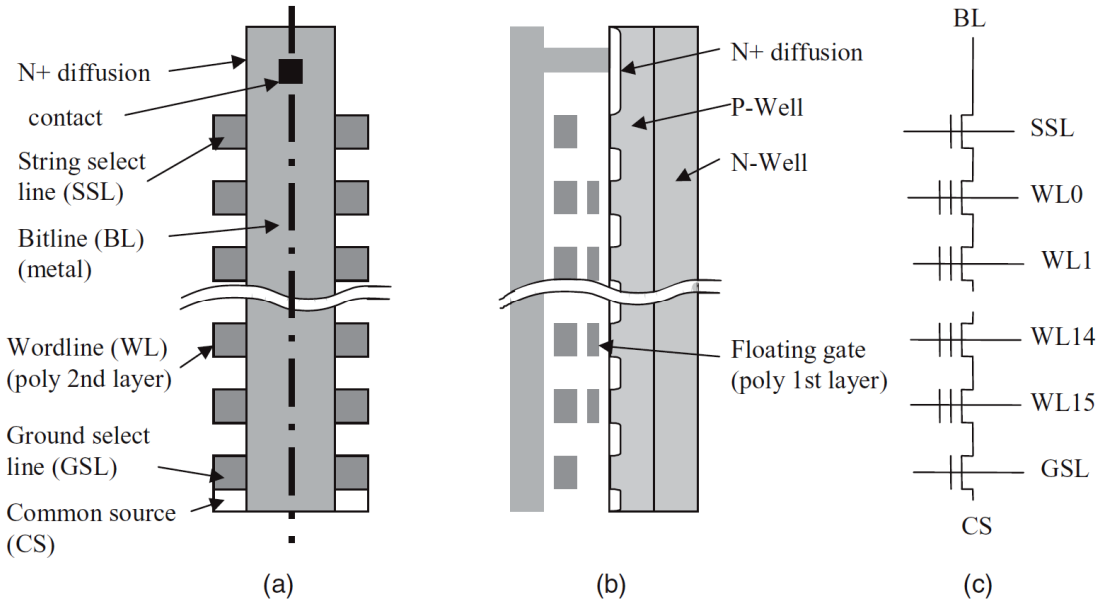


Fig. 3: Memoria Flash di tipo NAND

a) layout dell'array, b) sezione laterale, c) circuito equivalente. (Figura 3.6 di [4])

Le NAND sfruttano l'effetto tunnel di tipo Fowler-Nordheim sia per la programmazione sia per la cancellazione (nella tabella 1 sono riportate le tensioni tipiche di funzionamento).

Le memorie NAND hanno una maggiore complessità circuitale e sono meno affidabili delle memorie NOR e per tale motivo hanno bisogno di circuiti aggiuntivi che effettuano, laddove necessario, la correzione di eventuali errori nelle operazioni tramite tecniche di error correction code (ECC). Viene inoltre implementata una certa ridondanza delle celle alfine di effettuare delle "sostituzioni di celle" durante la vità della memoria e renderle così più robuste e tolleranti ai malfunzionamenti.

Tabella 1: Tensioni tipiche di utilizzo della memoria flash di tipo NAND di figura 2.

	BL	CS	Selected WL	Not Selected WL	GSL	SSL	PWell
Program	Ground (Data=0) VCC (Data=1)	5V	19V	12V	GND	5V	GND
Erase	Floating	Floating	GND	GND	Floating	GND	20V
Read	3V	GND	GND	VCC	VCC	VCC	GND

L'elevata densità di memoria e la minore dimensione delle celle permette alle memorie flash NAND di scrivere e cancellare più velocemente blocchi di dati, rendendole ideali per audio/video storage dove l'accesso ai dati è prevalentemente di tipo sequenziale. Le memorie Flash NAND sono dunque ideali per le applicazioni a basso costo, ad alta densità e ad alta velocità di programmazione/cancellazione, come ad esempio la memorizzazione per dispositivi di consumo, quali fotocamere digitali o unità disco USB.

La figura 4 riassume e mette a confronto le peculiarità delle due tipologie di memorie comparandone la capacità di memorizzazione, la velocità di lettura, la velocità di scrittura, il consumo di potenza in funzionamento ed in stand-by, il costo per bit e la tipologia di utilizzo.

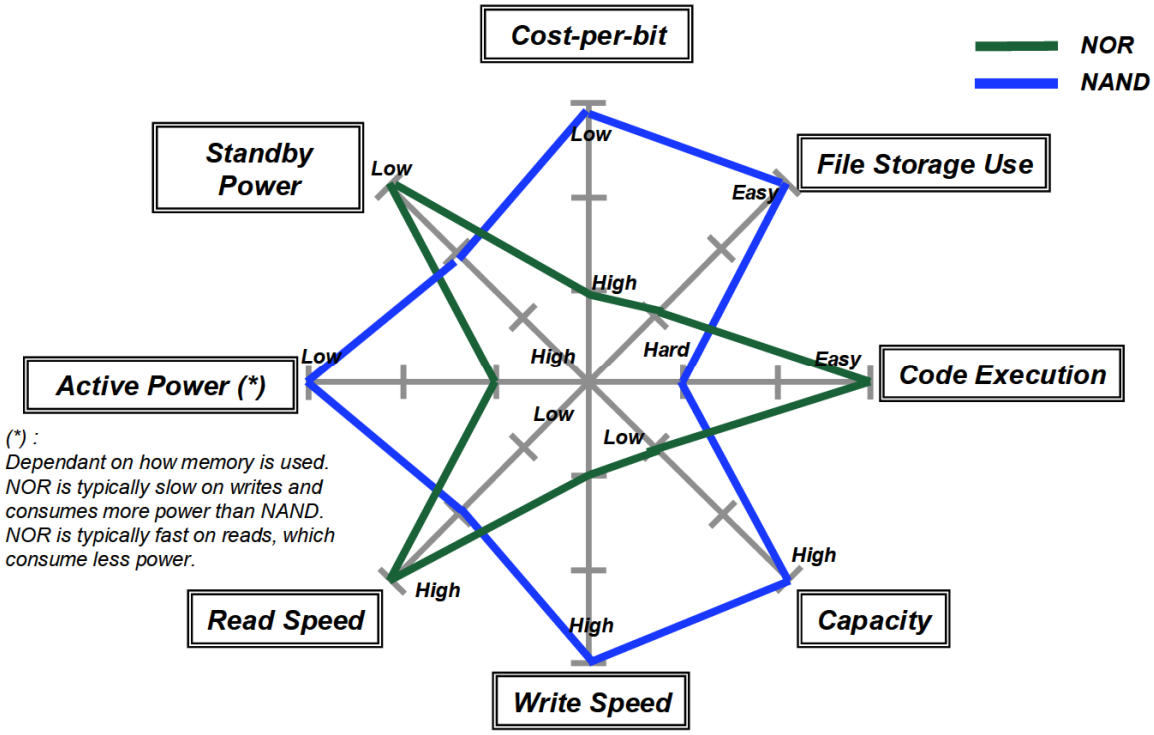


Fig. 4: Comparazione delle caratteristiche funzionali fra NOR e NAND Flash Memory.

(Figura 1 di [5])

Nella figura 5 viene invece riportato un confronto fra le caratteristiche operative e le performance delle memorie flash di tipo NOR e NAND e dunque vengono individuati i tipici campi di utilizzo delle stesse.

	SLC NAND Flash (x8)	MLC NAND Flash (x8)	MLC NOR Flash (x16)
Density	512 Mbits[1] – 4 Gbits[2]	1Gbit to 16Gbit	16Mbit to 1Gbit
Read Speed	24 MB/s[3]	18.6 MB/s	103MB/s
Write Speed	8.0 MB/s	2.4 MB/s	0.47 MB/s
Erase Time	2.0 mSec	2.0mSec	900mSec
Interface	I/O – indirect access	I/O – indirect access	Random access
Application	Program/Data mass storage	Program/Data mass storage	eXecuteInPlace

Fig. 5: Esempio delle caratteristiche operative delle memorie NAND e NOR. (Figura 4 di [5])

Nel prossimo paragrafo saranno affrontate con maggiore dettaglio le operazioni eseguite dalle memorie e gli effetti di stress ad essi collegati.

2.2 Programmazione – Operazione ed Effetti

Come già accennato nel paragrafo precedente sono due i principi fisici adoperati per la programmazione di una cella Flash in dipendenza della tipologia NOR o NARD: il channel hot electron injection (CHEI) [6], utilizzato nelle memorie di tipo NOR; e il Fowler-Nordheim (FN) tunneling [6], utilizzato nelle memorie di tipo NAND.

La programmazione tramite CHEI consiste nell'iniezione nel floating gate di elettroni caldi, cioè d'elettroni dotati di un'energia cinetica, imposta da un campo elettrico, superiore a quella termica del cristallo di silicio (ossia del substrato della cella in cui si forma il canale). Il meccanismo d'iniezione nell'ossido è mostrato nella figura 6, che riporta i livelli energetici del silicio, dell'ossido e del polisilicio, a seguito dell'applicazione di un potenziale positivo elevato al gate della cella, che a sua volta induce una differenza di potenziale tra il floating gate e il substrato di silicio.

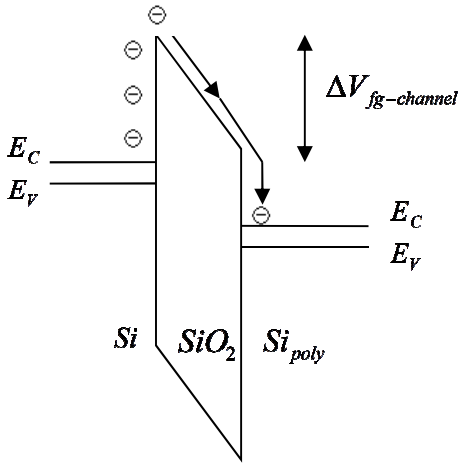


Fig. 6: Diagramma a bande substrato-ossido di tunnel-floating gate nella fase di programmazione per CHEI.

Tale configurazione dei livelli energetici permette il passaggio dal substrato al polysilicio dei portatori di carica con energia superiore alla barriera di potenziale dell'ossido. I potenziali tipicamente applicati, già accennati nel paragrafo precedente, sono riportati nella figura 7. Con queste tensioni di polarizzazione, la cella ha una corrente relativamente grande (0.3-1mA) e gli elettroni caldi generati nel canale acquisiscono un'energia sufficiente per saltare la barriera dell'ossido di gate e rimanere intrappolati nel floating gate.

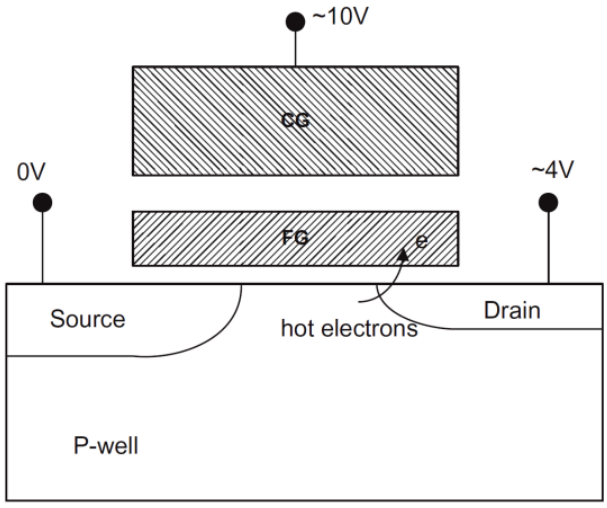


Fig. 7: Tensioni applicate per la programmazione CHEI.

Riassumendo, affinché gli elettroni possano essere iniettati nel floating gate devono essere soddisfatte tre condizioni:

- L'energia acquisita dall'elettrone deve essere tale da consentire di superare la barriera di potenziale costituita dall'ossido di tunnel che si trova tra il substrato e il floating gate;
- La quantità di moto dell'elettrone deve essere diretta verso il floating gate;
- Il control gate dovrà avere un potenziale superiore a quello del canale in modo da permettere al floating gate di essere un punto di raccolta per gli elettroni.

La scelta di applicare un'elevata tensione positiva al control gate consente di attirare maggiormente gli elettroni al floating gate, ma un valore troppo alto determinerebbe un eccessivo stress per l'ossido di tunnel. Inoltre, il transistor si porterebbe in zona di triodo, riducendo l'ampiezza della zona svuotata in corrispondenza della quale avviene la generazione d'elettroni caldi e diminuendo quindi l'efficienza di programmazione.

Dunque, il valore della corrente di gate (I_G) in funzione della tensione di gate (V_G) aumenterà ad un livello significativo solo quando il campo elettrico nell'ossido sarà sufficiente all'iniezione di elettroni. Successivamente diminuirà in funzione della corrispondente diminuzione del campo laterale con una tensione di gate crescente, questo fenomeno è evidenziato nella figura 8 che mostra un massimo di I_G intorno a $V_G=V_D$.

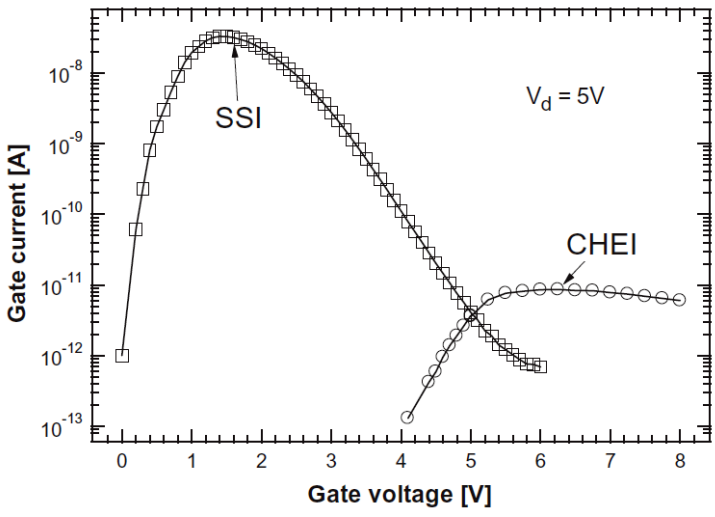


Fig. 8: Caratteristica IG vs VG a VD costante in fase di channel hot-electron injection.

La programmazione delle celle NOR per channel hot-electron injection è molto veloce, e si aggira tipicamente intorno ad 1µs per cella. Durante la programmazione di una cella tutte le altre celle che condividono con essa bit line e word line vengono disturbate dall'operazione di programmazione, nella figura 9 è mostrato una parte di array durante la programmazione di una cella.

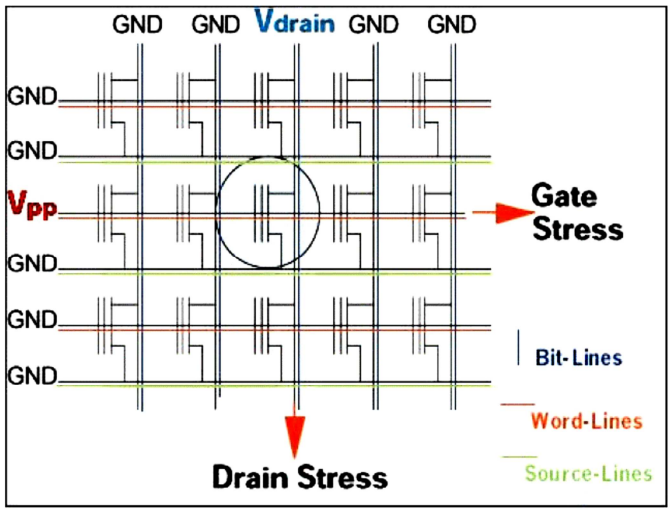


Fig. 9: Disturbi all'interno dell'array dovuto a programmazione.

L'alta tensione applicata sulla wordline può causare due diverse tipologie di fenomeni a seconda dello stato in la cella si trova, ossia se essa è cancellata o programmata. Nel caso di celle programmate gli elettroni tenderanno ad attraversare per FN lo strato di ossido d'interpolazione fra le due gate (ONO Oxide) per raggiungere la control gate (su cui sono applicati i 10V) tendendo quindi a sottrarre elettroni ed a cancellare la cella, mentre nel caso di celle cancellate gli elettroni tenderanno ad attraversare l'ossido sottile (Gate oxide) dal canale fino al floating gate tendendo in questo caso ad iniettare elettroni non voluti e quindi a programmare parzialmente la cella.

Entrambi i fenomeni sono definiti come gate stress e riducono i margini per le tensioni di soglia (basso e alto), come mostrato nella figura 10.

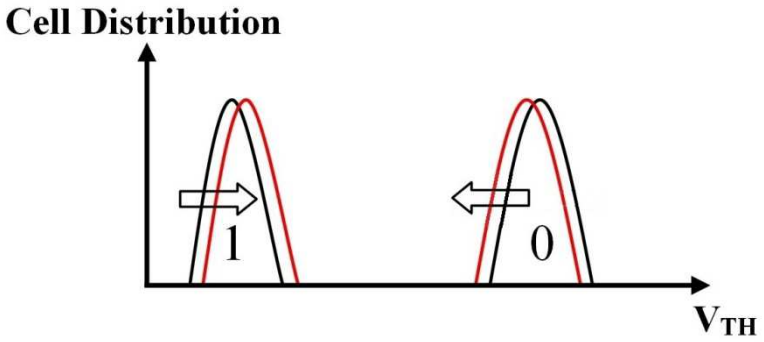


Fig. 10: Diminuzione dei margini di cancellazione e programmazione dovuti a gate stress nelle memorie flash NOR.

La scelta appropriata dell'architettura dell'array ed un'alta qualità dell'ossido di tunnel e del dielettrico fra gate e floating gate (interpoly) può garantire che sia ridotto al minimo possibile l'effetto di questo fenomeno, proteggendo dai cambiamenti non voluti lo stato delle celle non interessate da operazioni di programmazione. Questo è uno dei motivi per cui il dielettrico interpoly è spesso costituito da tre strati: biossido di silicio - nitruro di silicio - biossido di silicio (ONO Oxide-Nitride-Oxide). Il dielettrico multistrato cerca di impedire i problemi di generazione di cariche nell'ossido, il nitruro con la sua superiore costante dielettrica consente di utilizzare uno strato dielettrico più spesso senza influenzare l'accoppiamento fra il floating gate e il control gate. Ovviamente questa tipologia di fenomeno, provocando comunque usura dell'ossido, ha un impatto sulla resistenza a lungo termine della cella (endurance) e sul tempo di vita della memoria stessa, finendo per generare fenomeni di fail della memoria.

Le celle sulla stessa bit line vedono la tensione di drain di programmazione (~5V) ed hanno la loro control gate ancorata al potenziale di massa; in queste condizioni esse vedono dunque 5V applicati che effetto tunnel possono spostare cariche dal floating gate attaverso l'ossido al drain. Inoltre, nel substrato possono essere generate, per ionizzazione da impatto, delle lacune che a sua volta possono essere iniettate nel floating gate; questa ulteriore perdita di carica si traduce in uno spostamento (shift) delle tensioni di soglia verso sinistra per le celle programmate. Le celle cancellate invece potrebbero guadagnare alcune cariche per effetto di Hot Electron Injection, l'effetto del drain stress nelle celle cancellate è minore e non desta storicamente preoccupazione. Gli effetti sulle distribuzioni delle tensioni di soglia sono mostrati in figura 10.

Nelle memorie Flash di tipo NAND la programmazione avviene invece per effetto di Fowler-Nordheim (FN), con un meccanismo di tunneling a campo assistito. Quando viene applicata una tensione elevata su una struttura Polisilicio-SiO_2-Silicio, il suo diagramma a bande viene modificato a causa dell'alto campo elettrico applicato, come mostrato in figura 11. In queste condizioni, gli elettroni vedono una barriera triangolare la cui larghezza dipende dal campo applicato, l'altezza è determinata dal materiale dell'elettrodo e dalla struttura a banda dell'ossido (SiO_2). Per campi sufficientemente alti e spessori dell'ossido di tunnel bassi, la larghezza della barriera di potenziale diventa sufficientemente piccola ed alcuni elettroni possono attraversare la stessa per effetto tunnel. Questo meccanismo è stato dimostrato da Fowler e Nordheim per il caso di tunneling di elettroni attraverso una barriera a vuoto ed è stato poi descritto anche da Lenzlinger e Snow per il tunneling attraverso ossidi [7].

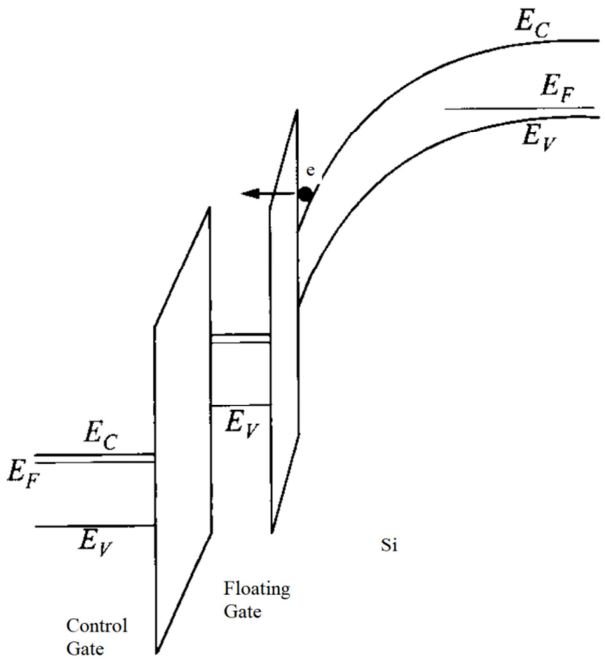


Fig. 11: Fowler-Nordheim tunneling in fase di program delle memorie flash NAND.

Per effettuare l'operazione di programmazione per effetto tunnel nelle memorie flash NAND vengono applicate tensione dell'ordine dei 20V fra control gate e substrato, come mostrato in figura 12.

In particolare per la programmazione di uno zero (0-program), il source e il drain della cella vengono portati a massa e gli elettroni vengono iniettati per FN dallo strato di inversione fino al floating gate, di conseguenza la tensione di soglia della cella di memoria cambia dal valore negativo dello stato cancellato al valore positivo.

Invece, per la programmazione di un uno (1-program - inibith) il source e drain della cella vengono portati a 8V, in questo caso la tensione di soglia della cella di memoria è mantenuta nello stesso stato perché la tensione sopra l'ossido di tunnel e sotto il floating gate è troppo bassa per causare una corrente di tunnel. Le celle selezionate vengono portate ad una tensione di gate (word line) pari a circa 20V, mentre le wordline non selezionate sono portate a circa una decina di volt.

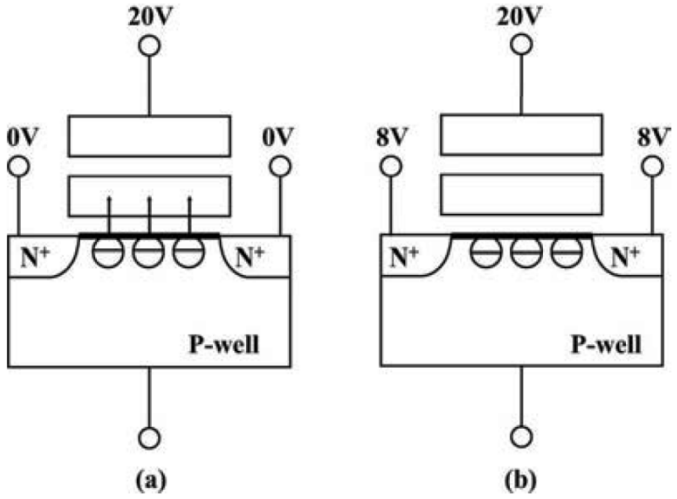


Fig. 12: Tensioni applicate in fase di programmazione per FN tunneling alle celle flash NAND.

a) 0-program b) 1-program.

Durante la programmazione di una determinata cella di memoria all'interno dell'array, le celle non selezionate subiscono dei disturbi dovuti all'elevata tensione applicata tra il canale delle celle e il floating gate. Come mostrato nella figura 13 esistono due tipologie di disturbo durante la programmazione di una cella, perché alle celle che non devono essere programmate (Cell B in figura 13) ma stanno sulla stessa wordline è applicata una tensione di gate di 20V ed una tensione di bitline pari a 7V; mentre alle celle sulla stessa bitline (Cell A in figura 13) è applicata una tensione di wordline pari alla tensione delle celle non selezionate (V_M) e una tensione di bitline pari a 0V.

E' per tale motivo che il livello della tensione V_M scelta deve essere impostato per tener conto di queste due condizioni ed in definitiva cercare di minimizzare gli effetti di stress sulle celle non selezionate.

Se ad esempio esso fosse posto al valore massimo, pari a 20V, Cell A vedrebbe dei potenziali ottimi per la programmazione per effetto tunnel; se invece il potenziale V_M fosse posto ad un valore inferiore a quello tale che V_M-Vth_0<7V (Vth_0=tensione di soglia della Cell B), la tensione fra il control gate e il canale sarebbe 20V-(VM-Vth_0) e dunque la Cell B potrebbe subire una programmazione per effetto tunnel. I risultati sperimentali, figura 13, mostrano che il valore di VM ha un campo nel quale è possibile minimizzare lo stress provocato dai due effetti e i valori tipici stanno nel campo 8÷10V.

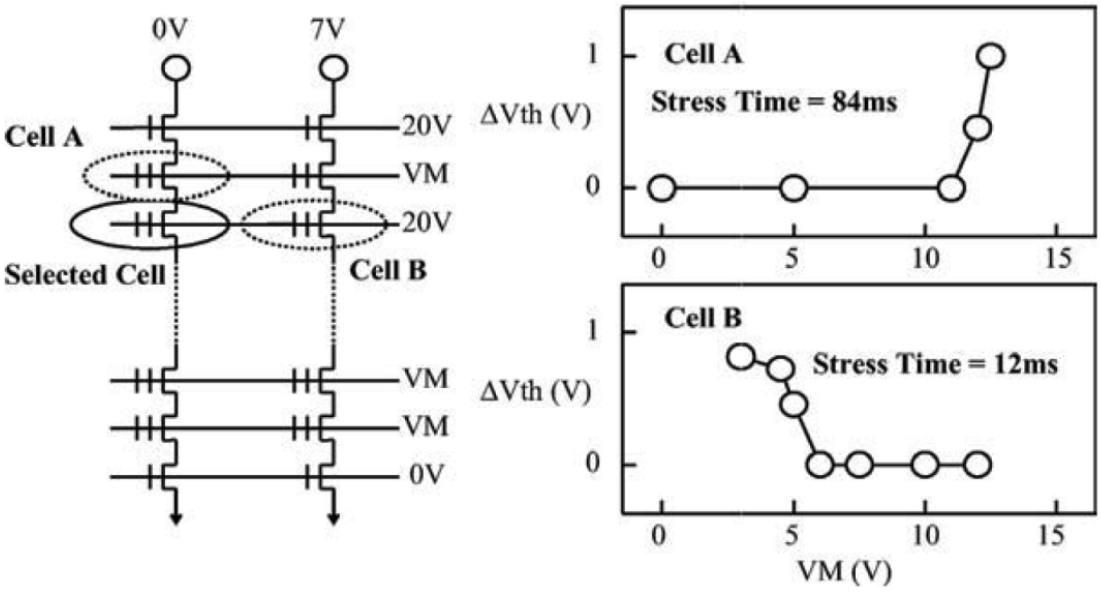


Fig. 13: Stress subiti dall'array NAND durante l'operazione di program.

2.3 Cancellazione – Operazione ed Effetti

L'operazione di cancellazione avviene per Fowler-Nordheim tunneling sia per l'architettura NOR che per quella NAND e le due architetture differiscono soltanto per le tensioni applicate.

Il meccanismo del FN tunneling è stato descritto nel paragrafo precedente, il diagramma a bande relativo alla operazione di cancellazione è mostrato nella figura 14.

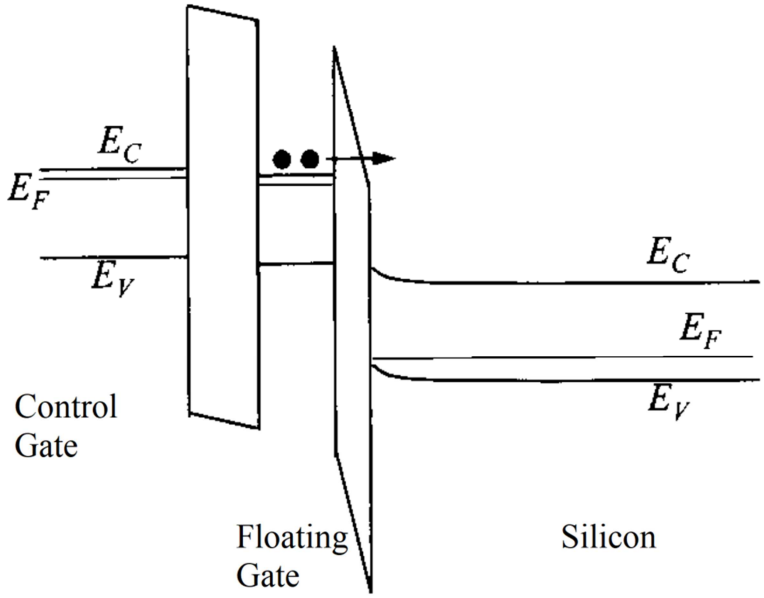


Fig. 14: Diagramma a banda del Fowler-Nordheim tunnelling in fase di cancellazione.

Nell'architettura NOR la cancellazione per FN tunneling può essere ottenuta con differenti configurazioni di tensioni applicate ai terminali del blocco o della pagina interessata, come mostrato nella figura 15. In particolare, figura 15 a), l'effetto tunnel può avvenire fra la floating gate ed il terminale di source applicando una tensione elevata (~11V) al source, lasciando il drain flottante e portando a massa la control gate. In alternativa è possibile portare ad un potenziale negativo il control gate (~-10V) con il source a circa 4V ed il drain sempre flottante.

Un'altra metodologia per la cancellazione delle memorie flash NOR è mostrata in figura 15 b). Questa risulta essere più similare alla cancellazione delle memorie Flash di tipo NAND. Tale cancellazione avviene per FN tunnelling dal floating gate verso il substrato applicando una tensione positiva al substrato, di circa 7V, ed una tensione negativa al control gate, di circa -8V. Nelle memorie flash NOR la cancellazione per FN attraverso il canale è possibile solo se è possibile applicare una tensione positiva al substrato.

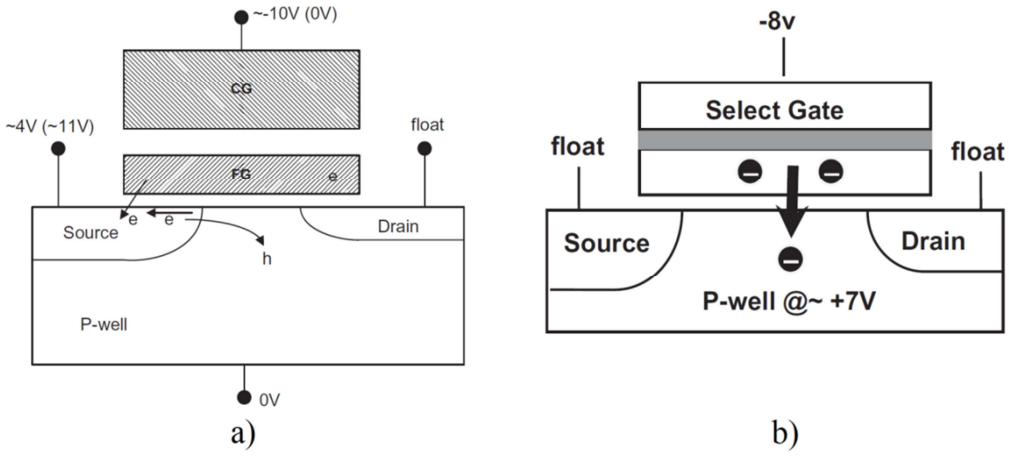


Fig. 15: Tensioni applicate per la cancellazione delle celle flash NOR.

a) FN tunneling fra floating gate e source. b) FN tunneling fra floating gate e substrato.

La prima polarizzazione descritta per la cancellazione è comunemente indicata come "cancellazione con source positivo", la seconda è nota come "cancellazione a gate negativa" e la terza è conosciuta come "cancellazione tramite canale". Il vantaggio della polarizzazione con gate negativa è la capacità di realizzare l'operazione con l'uso di tensioni di source relativamente più basse (solo 4V invece degli 11V); ciò permette a sua volta una più semplice ingegnerizzazione della giunzione che deve in questo caso sopportare minori tensioni massime di funzionamento rispetto alla cancellazione a source positivo. Un problema tipico dell'architettura NOR nella cancellazione è la così detta sovra-cancellazione, cioè l'ottenimento di celle deplete (a V_{TH} negativa). In questo caso infatti una wordline non selezionata (e quindi polarizzata a 0V) sarà in grado di portare una cella depleta in conduzione causando una corrente indesiderata sulla colonna; tale corrente quindi può essere considerata come una sorgente di rumore sul segnale di lettura. Se molte celle in una colonna conducono per sovra-cancellazione, le loro correnti si accumuleranno e possono facilmente provocare un errore nella lettura della cella selezionata.

Per evitare questo problema si adottano due diversi metodi, il primo è quello di programmare l'intero blocco/pagina prima di effettuare la cancellazione evitare così la sovra-cancellazione. Il secondo metodo è invece successivo all'operazione stessa di cancellazione: in questo caso infatti, una volta ultimata la procedura di cancellazione, tutte le celle con un valore di V_{TH} relativamente basso vengono leggermente riprogrammate (soft-programming). La soft programmazione è una procedura identica alla programmazione per CHEI, ma in essa la tensione di gate è mantenuta ad un livello inferiore. Questa operazione è conosciuta anche come Post-Erase-Repair (PER).

Le tensioni applicate nell'architettura NAND per avere la cancellazione tramite FN tunneling sono riportate nella figura 16. Il substrato è polarizzato a circa 20V, mentre il control gate è posto a massa, ciò provoca FN tunneling, come già descritto in precedenza. La cella di memoria è volutamente sovra-cancellata a circa -3÷-2V perché l'architettura NAND tollera la sovra cancellazione senza un effetto nelle operazioni successive di lettura.

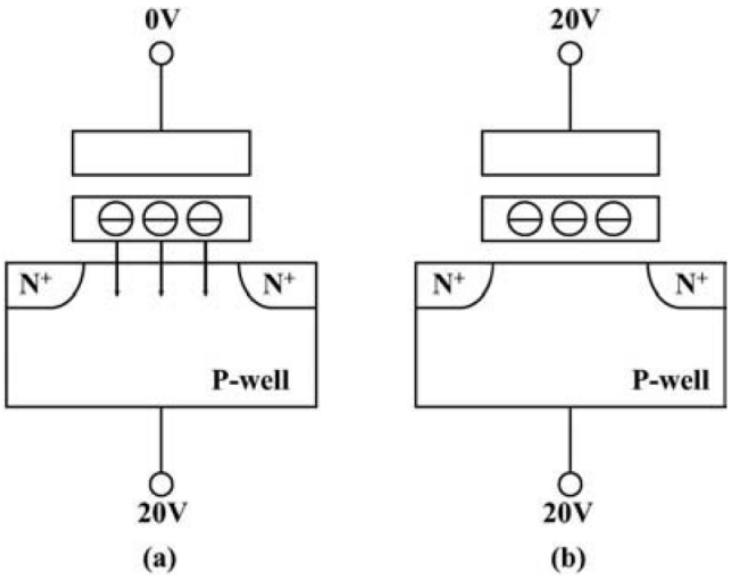


Fig. 16: Tensioni applicate durante la cancellazione per FN tunneling nelle memorie flash NAND.

a) Cancellazione b) Inibizione alla cancellazione

E' utile evidenziare che per evitare che la sacca P (P-well) della circuiteria periferica sia sottoposta alle tensioni elevate delle operazioni usuali di cancellazione su chip, il suo P-well è isolato da quello delle celle costituenti l'array di memoria.

Nei blocchi di memoria che non devono essere cancellati viene realizzato il collegamento del control gate allo stesso potenziale del substrato, così da inibire la cancellazione, come riportato in figura 16-b. Nell'operazione di verifica della cancellazione, tutte le linee word del blocco selezionato sono impostate su 0 V, consentendo così la verifica simultanea di tutte le celle.

2.4 Lettura – Operazione ed Effetti

Per leggere una cella di memoria flash bisogna applicare un opportuno valore di tensione di gate, superiore al valore massimo della distribuzione di soglie delle celle cancellate, in modo da rilevare la corrente sulla bitline e confrontarla con una relativa alla cella di riferimento, valutando così se la cella sia stata in precedenza programmata a livello logico '0' o '1'. La tensione di gate di lettura è differente per le due architetture, flash NOR e NAND.

La lettura nell'architettura NOR viene effettuata applicando al terminale di control gate della cella interessata una tensione di circa 5V, tenendo source e substrato a massa e con una tensione di drain di circa 1V, così da permettere l'eventuale flusso della corrente.

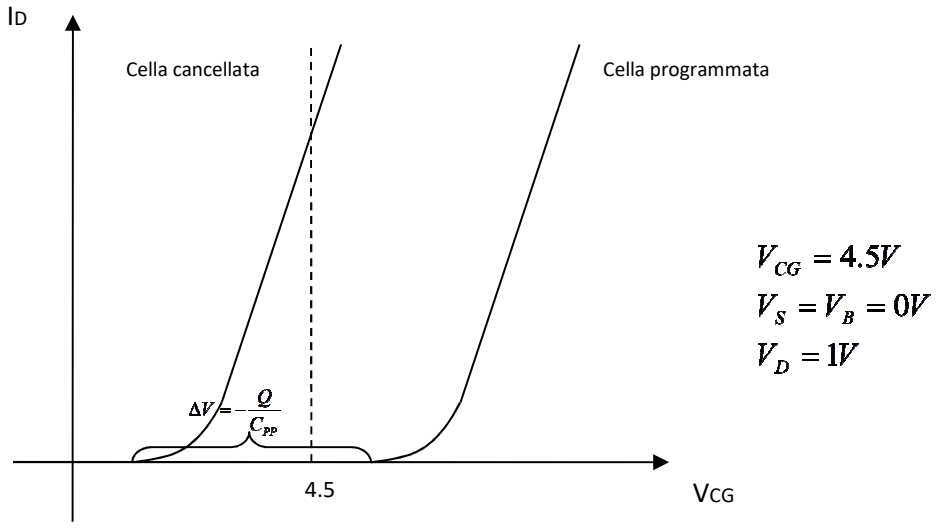


Fig. 17: Caratteristica di una cella programmata (0) e di una cancellata (1) per array di tipo NOR.

Nella fase di lettura le tensioni applicate non sono così elevate da produrre in brevi tempi errori o disturbi, ma l'elevato numero di letture può provocare effetti cumulativi di gate stress nelle celle poste sulla stessa wordline della selezionata, l'effetto del gate stress è stato mostrato precedentemente in figura 9.

Questi effetti sono tipicamente trascurabili date le basse tensioni, si possono comunque prevedere algoritmi di refresh per prevenire la perdita di carica.

La lettura nell'architettura NAND viene effettuata applicando le tensioni riportate in figura 18, in particolare sono applicati 5V a tutti i control gate delle celle della stessa bit line, ad eccezione di quella selezionata, che è collegata a massa. I transistori di selezione sono in conduzione e collegano la serie di celle alla massa e alla bitline. La bitline viene pre-caricata, se i dati memorizzati sono "1" (tensione di soglia negativa) e il transistore selezionato è conduttivo e scarica la linea di bit, se i dati memorizzati sono "0", il transistor è spento e la linea di bit mantiene la carica.

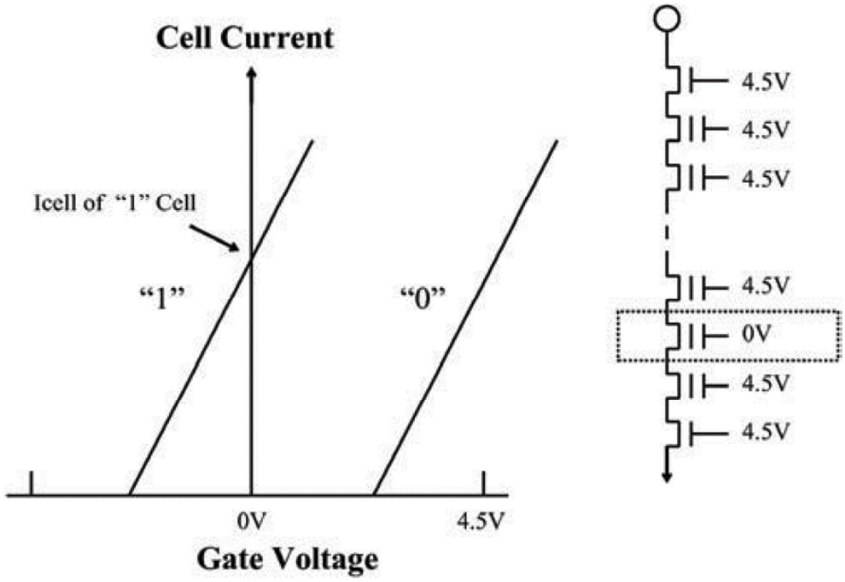


Fig. 18: Caratteristica di una cella programmata (0) e di una cancellata (1) per array di tipo NAND.

In questa tipologia di architettura tutte le word line di un blocco tranne quella selezionata sono poste a Vread (4,5V) e quindi tutte queste celle subiscono in linea di principio un effetto di gate stress. Tali effetti sono stati studiati e risultano praticamente di valore trascurabile rispetto allo stress che le celle subiscono in fase di cancellazione. Essi quindi non determinano alcun deterioramento percettibile nella determinazione della vita di una memoria.

2.5 Test con Automated Test Equipment (ATE)

Il test delle memorie Flash viene effettuato in fase di produzione, così come per tutti i circuiti integrati, tramite delle apparecchiature all'uopo progettate e denominate Automated Test Equipments. Tali test consistono in una serie di procedure per la generazione di segnali di stimolo che vengono forniti ai chip sotto esame onde rilevare il loro comportamento sia in condizioni di operatività normale, sia forzando in valori di tensione a livelli differenti onde poter captare potenziali vulnerabilità dei chip e determinare se vi siano punti deboli che potrebbero portare i chip stessi ad avere una vita breve e successivamente determinare dei fallimenti non voluti. Tali procedure sono di norma sviluppate ad hoc per ogni specifico dispositivo, altresì detto Device Under Test (DUT).

Data la complessità circuitale delle memorie flash, sicuramente definibili come circuiti di tipo mixed-signal, esistono un gran numero di possibili tipologie di guasto e questo si traduce nella realizzazione di algoritmi di test complessi e dai tempi di esecuzione lunghi. Tale lunghezza dei tempi di prova si traduce in un aggiunta di costo elevato per la produzione ed in definitiva in un aumento del costo del singolo chip.

D'altro canto il costo di un'apparecchiatura ATE per le memorie flash è tipicamente funzione: del numero di dispositivi contemporaneamente testabile (grado di parallelismo) e della sua frequenza di funzionamento. Infatti maggiore sarà la velocità di esecuzione e minore sarà il tempo impiegato per il test del singolo chip.

Il flusso di testing sviluppato su ATE per le memorie flash è diviso in Front End o ciclo di EWS (Electrical Wafer Sort) e in Back End o test finale, come si evince nella figura 19. Il ciclo di EWS viene diviso normalmente in due parti con una fase di bake per la verifica della ritenzione dei dati tra una parte del test e l'altra (alcune applicazioni prevedono anche una fase detta di module testing, se per la memoria è previsto il montaggio in un modulo specifico).

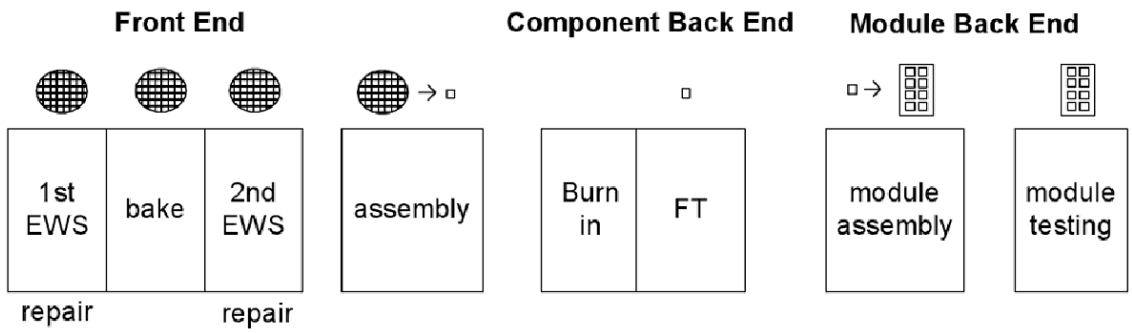


Fig. 19: Flusso di test previsto per le memorie flash utilizzando ATE.

Tipicamente il primo test EWS è utilizzato per valutare malfunzionamenti immediatamente individuabili, detti anche difetti precoci, e per testare la funzionalità dei blocchi. Ad esempio per l'individuazione di corto circuiti o malfunzionamenti delle wordline o delle bitline che vengono sostituite con le wordline o le bitline ridondanti, già previste in fase di design.

Un flusso del primo EWS (EWS1) è normalmente così strutturato:

- DC e contact test;
- Test sulla funzionalità dei buffer;
- Ciclatura iniziale;
- Test di Cancellazione e Lettura;
- Primo test per l'individuazione di difetti;
- Test di programmazione e lettura;
- Test sulle tensioni;
- Stress test per l'attivazione di eventuali difetti;
- Test sulla programmazione Checkerboard/Inverse Checkerboard e verifica;
- Sommario delle eventuali ridondanze e l'attivazione dei nuovi percorsi tramite fusibili.
- Pre-condizionamento tramite forno.

I wafer che superano l'EWS1, vengono inviati al forno (bake) e tale procedura avviene a temperature fra i 150°-250°C per 24/48 ore. Essa viene utilizzata per identificare i guasti di ritenzione dovuti a meccanismi con energia di attivazione più elevata di 0,6eV.

Successivamente, si svolge il secondo EWS (EWS2) che, tra i vari test, effettua anche la verifica della ritenzione dei dati nelle celle dell'array.

Il flusso tipico del secondo EWS è il seguente:

- DC e contact test.
- Lettura dei risultati post bake.
- Scrittura di un pattern diagonale per verificare eventuali difetti all'interno dei decoder.
- Test di utilizzo in modalità utente.
- Sommario delle eventuali ridondanze e attivazione dei dei nuovi percorsi tramite fusibili.

I wafer che superano l'EWS2 sono inviati alla procedura detta di burn-in. Tale procedura consiste in una serie di operazioni di tipo program-verify-erase-verify eseguiti ad alta temperatura (125°C), individuando in tal modo possibili fail precoci e dunque cercando di diminuire il failure che si potrà avere nel funzionamento in modalità utente.

Il flusso di test finale, o di back-end, serve a garantire che i parametri del dispositivo rientrino nelle specifiche di progetto. Inoltre, questo test serve anche ad individuare eventuali fallimenti/difetti introdotti dal confezionamento (assembly) del prodotto ed in questa fase tutte le misure sono effettuate a bassa temperatura.

Le fasi principali del test finale sono le seguenti:

- Misure delle correnti di dispersione (leakage), di stand-by e di funzionamento.
- Verifica del funzionamento all'interno di tutto il range di alimentazione.
- Test specifici per la stabilità del dispositivo.
- Verifica dei parametri al variare della velocità di funzionamento.
- Lettura del chip ID per la tracciabilità e l'immagazzinamento dei risultati delle misure per le analisi di resa dei dispositivi (Yield).

Le architetture degli ATE utilizzate per i flussi di testing sopra descritti si possono dividere in due categorie, figura 20:

1) Architettura di tipo parallelo, in cui l'ATE riserva ad ogni memoria in fase di testing (DUT) un hardware completamente dedicato per l'intervallo di tempo di test necessatio. Elementi fondamentali di tale hardware sono il Test Site Controller (TSC), l'Algorithm Pattern Generator (APG), il Parametric Measurement Unit (PMU), il Vector Memory (VM), il Programmable Power Supply (PPS), l'Error Catch RAM (ECR) ed il Data Buffer Memory (DBM). In questa modalità il flusso di test viene svolto in modo asincrono fra i vari DUT presenti nell'ATE.
2) Architettura condivisa: in cui si ha un solo hardware che viene condiviso da più DUT, dunque il test da parte dell'ATE deve essere svolto in modo sincrono fra i vari DUT.

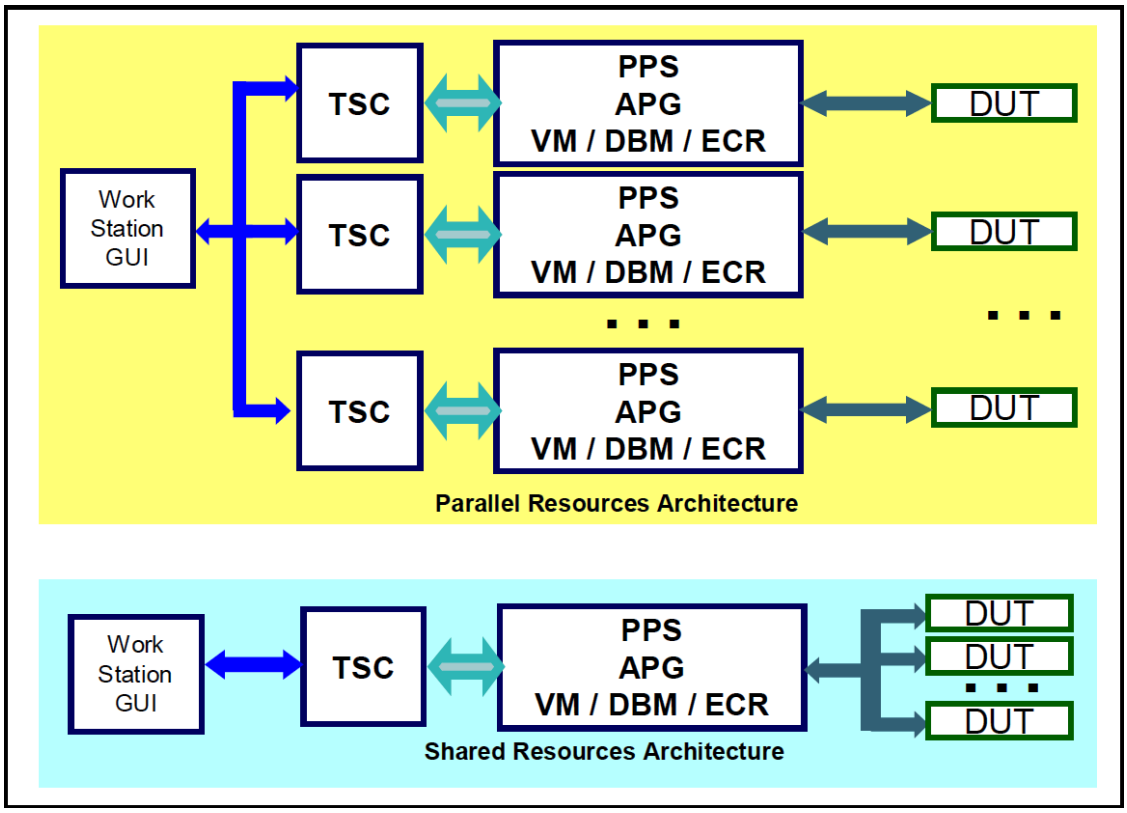


Fig. 20: Architetture degli Automated Test Equipment (ATE).

2.6 Built In Self Test (BIST)

Negli ultimi decenni il testing classico con ATE è stato affiancato dalla nascita di tecniche, denominate BIST (Built-In-Self-Test), tramite le quali si prevede in fase di progetto l'inclusione dei requisiti di osservabilità e controllabilità del chip dando così origine a quella che sotto il nome di Design for Testability (DfT) [7, 8]. Tale metodologia comporta l'inclusione nel chip di componenti hardware e software supplementari per permettere l'auto-test (funzionale e/o parametrico), riducendo parzialmente in tal modo la dipendenza delle fasi di test da apparecchiature ATE esterne.

In tal senso quindi si impiega un maggior quantitativo di risorse nel chip, sia in termini di tempo di progettazione sia in termini di area di silicio occupata dal chip finale, alfine di ridurre in modo consistente il tempo di esecuzione del test completo compensando in tal modo l'aggravio di spesa con una conseguente e talvolta conveniente riduzione dei costi della fase di test. In alcuni casi ciò migliora l'accuratezza delle prove e soprattutto permette la realizzazione di test del circuito integrato in qualsiasi istante e condizione di funzionamento (in-field test), cosa ovviamente impossibile con gli ATE convenzionali, a meno di un rientro in fabbrica dei chip già posti su mercato.

Il DfT è una disciplina che tenta di migliorare le caratteristiche del test cambiando il modo in cui i progettisti si avvicinano al suo sviluppo ed un breve quadro riassuntivo delle metodologie di DfT è mostrato nella figura 21.

L'obiettivo progettuale, come detto, è quello di aumentare la controllabilità e l'osservabilità dei nodi all'interno del circuito, a tal fine sono necessarie metodologie quali scansioni a catena e schemi di testing casuali e/o partizionati. Il successo dell'approccio DfT viene misurato dalla capacità di generare un set di test per il DUT che produca un'elevata e stabile copertura dei guasti.

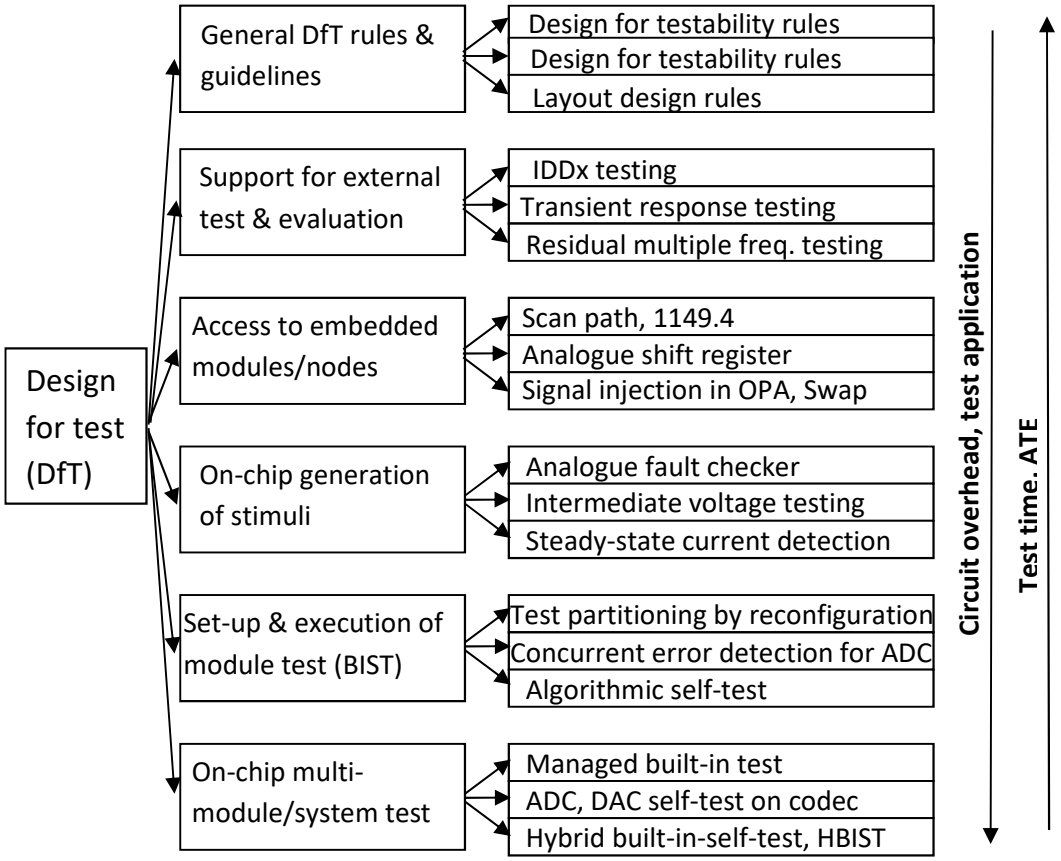


Fig. 21: Sintesi dei metodi Dft.

Le tecniche DfT discusse sopra possono essere impiegate in varie combinazioni per implementare test completi all'interno dei circuiti VLSI e fra queste le tecniche BIST [9, 10, 11] sono quelle maggiormente utilizzate per il test delle memorie flash, l'architettura base del BIST è mostrata in figura 22.

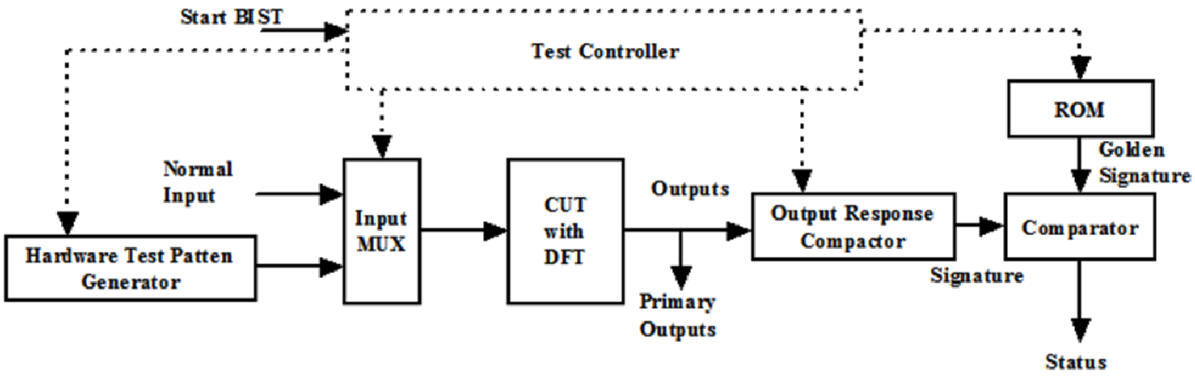


Fig. 22: Generica architettura BIST.

Come si evince dalla figura 22, il generico circuito BIST è formato da:

1. Hardware Test Pattern Generator (TPG), questo modulo genera i test pattern utilizzati per l'individuazione dei guasti/errori e per la propagazione degli stessi alle uscite. Esso è fondamentalmente un registro che genera pattern casuali. La caratteristica principale del TPG è quella di avere un piccolo impatto in termini di area occupata, ma generare un numero molto elevato di pattern diversi (utilizzando n flip-flop si riescono a realizzare fino a 2^n-1 pattern diversi).
2. Input multiplexer (Input MUX), questo è il multiplexer d'ingresso permette la scelta fra gli ingressi per il normale utilizzo e gli ingressi provenienti dal TPG, per l'effettuazione del test.
3. Output Response Compactor, questo circuito digitale esegue la compressione delle uscite del circuito sottoposto al test. Tali uscite vengono compresse per rendere più semplice il successivo confronto con i risultati attesi, necessario per poter individuare gli eventuali guasti.
4. ROM, questa immagazzina i risultati attesi che si utilizzano per la comparazione.
5. Comparator, struttura hardware per la comparazione dei risultati reali-attesi che genera l'eventuale status di fallimento.
6. Test Controller, questo è il circuito che controlla il BIST dopo che lo stesso viene abilitato.

Il fulcro della struttura del BIST è il Test Pattern Generator, questo deve occupare la minor area possibile e generare pattern pseudo-esaustivi. La struttura più comune per la realizzazione del TPG è il Linear Feedback Shift Register (LFSR) perché soddisfa le due condizioni precedenti. Gli LFSR maggiormente utilizzati in questo contesto sono denominati standard e modulare.

La rappresentazione circuitale del LFSR standard è mostrata in Figura 23. L'LFSR, come detto in precedenza, è fondamentalmente un registro a scorrimento dotato di flip-flop D dove l'uscita dell'ultimo flip-flop fornisce feedback all'entrata del primo flip-flop. Se ci sono n flip-flop (numerati come X_0, X_1, X_2, …X_n), allora questo viene chiamato ad n-stage. Il feedback è normalmente realizzato tramite semplici porte logiche di tipo XOR che intercettano le uscite dei flip-flop. Nella rappresentazione circuitale visibile in Figura 23 la retroazione dei segnali dei flip flop, imposta dalle porte XOR, è inclusa o esclusa in dipendenza del valore h_i=0/1 deciso dal TPG, determinando in tal modo molti pattern differenti.

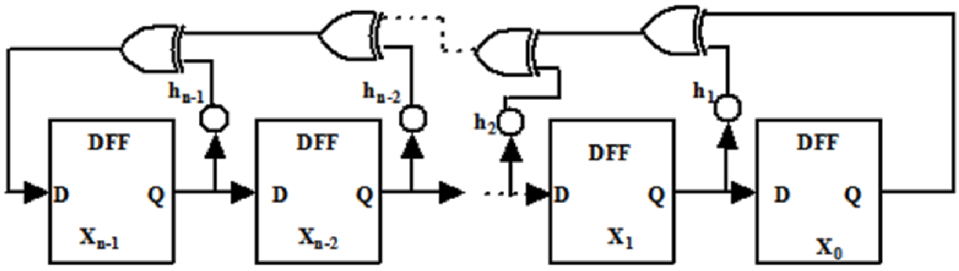


Fig. 23: Rappresentazione circuitale dell'LFSR standard.

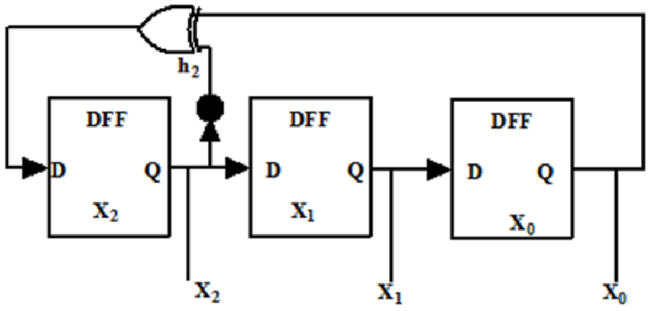


Fig. 24: Esempio di LFSR standard a tre bits.

Nella figura 24 è rappresentato un esempio di LFSR a 3 bit. Se i valori iniziali dei flip-flops sono: X_0=1; X_1=0; X_2=0. La sequenza dei pattern sarà la seguente:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Quindi il dispositivo genera, partendo da 3 bit, 7 pattern diversi (viene escluso quello con i bit tutti a zero), successivamente i pattern vengono ripetuti.

La rappresentazione circuitale del LFSR modulare è mostrata invece in Figura 25. A differenza del precedente, nel LFSR modulare è la posizione delle porte XOR a determinare la funzione di feedback; in quest'ultimo caso infatti le porte XOR sono poste tra un flip-flop ed un altro. Tale architettura risulta esser più veloce dato che nella catena di feedback si trova al massimo una porta XOR, mentre nella configurazione standard possono esserci diversi livelli di porte XOR.

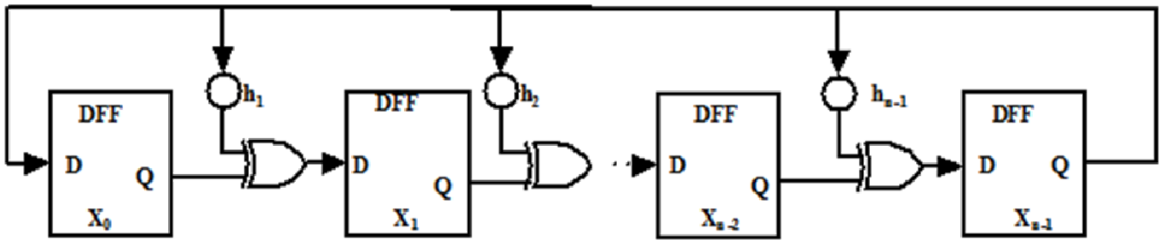


Fig. 25: Rappresentazione circuitale dell'LFSR modulare.

Nella figura 26 è mostrato un esempio di LFSR modulare a quattro bit.

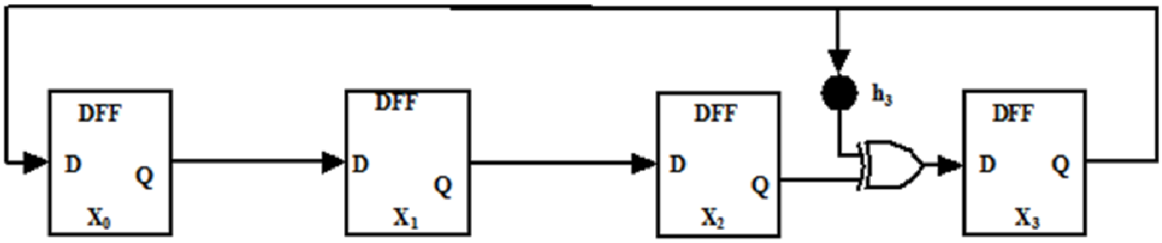


Fig. 26: Esempio circuitale di un LFSR modulare a 4 bits.

Se i valori iniziali dei flip-flop sono: X_0=1; X_1=0; X_2=0; X_3=0. La sequenza dei pattern sarà la seguente:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Quindi il dispositivo partendo da 4 bit genera 15 pattern diversi (viene escluso quello con i bit tutti a zero), successivamente i pattern vengono ripetuti.

Un ulteriore miglioramento delle tecniche di BIST si è avuto con lo sviluppo delle cosiddette tecniche di BIST Trasparente [12,13]. Queste sono state sviluppate per rispondere alla necessità di effettuare il testing in-field della memoria, come ad esempio all'accensione o durante i periodi di inattività (idle), cioè durante l'uso normale da parte dell'utente finale. Tali tecniche permettono infatti la conservazione dei dati in memoria, agendo così in modo assolutamente trasparente all'utente.

2.7 Software Based Self-Test (SBST)

Negli ultimi anni è stato effettuato un ulteriore passo in avanti nel testing in-field grazie allo sviluppo di alcune tecniche denominate SBST (Software-Based Self Test). Queste sono state sviluppate, a partire dagli anni ’80, per il testing dei microprocessori e sono oggi parte integrante del flusso di test dei maggiori produttori di circuiti integrati[14]. Un esempio di flusso di testing è riportato in figura 27, in cui si evince il ruolo supplementare degli SBST all’interno del flusso classico di test di un circuito integrato.

All’interno del wafer test esso implementa dei test strutturali per verificare la funzionalità del circuito; nel package test l’SBST viene inserito per aumentare il failure coverage all’interno degli altri test funzionali/parametrici.

Le tecniche SBST fanno sostanzialmente uso dell’intelligenza locale del microprocessore contenuto nel chip per condurre i test normalmente supportati da un intelligenza esterna, come nel caso degli ATE, o di hardware aggiuntivo, come nel caso delle tecniche BIST.

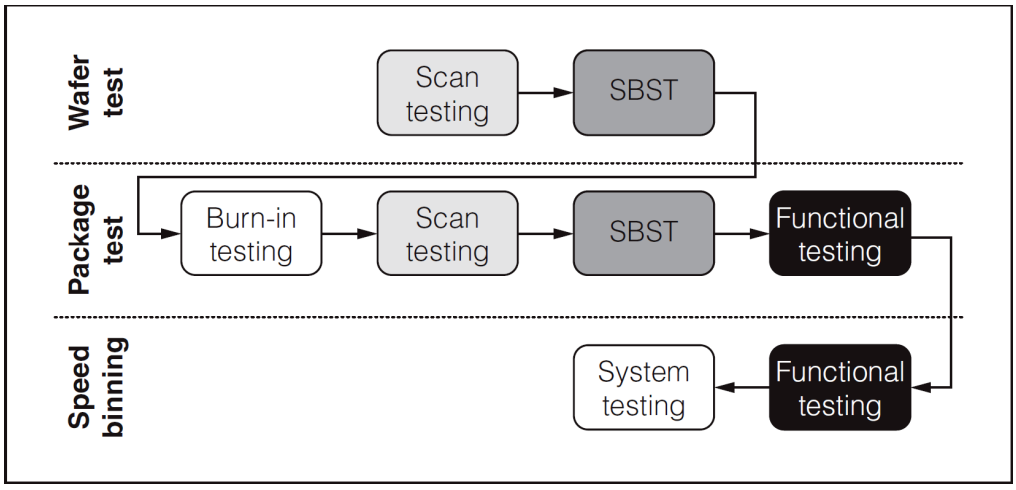


Fig. 27: Tipico flusso di testing dei microprocessoriper alti volumi di produzione. (figura 2 di [14])

Le caratteristiche dei software based self-test (SBST) sono molteplici:

- è una tipologia di test non intrusiva. Essa infatti non necessita di hardware aggiuntivo, così come accade nel caso di presenza di strutture BIST, e questo talvolta può risultare inaccettabilmente costoso per circuiti già complessi ed accuratamente ottimizzati come nel caso di microprocessori.
- Sono test normalmente molto veloci. Essi vengono eseguiti alla velocità effettiva del processore consentendo inoltre lo screening di difetti dovuti a ritardi temporali (delay) normalmente non individuabili a basse velocità.
- In-field testing. Come già evidenziato la loro caratteristica principale è possibilità d'esecuzione sul campo (cioè con dispositivo già in possesso dell'utente).

Le tecniche SBST hanno dimostrato notevole flessibilità di impiego ed una totale copertura dei test precedentemente effettuati tramite BIST, con il solo aumento del tempo di testing complessivo ma senza necessità di hardware aggiuntivo. Nell'ultimo decennio le tecniche SBST stanno trovando ulteriore applicazione nei System on Chip (SoC), in particolare sono stati effettuati diversi studi per l'applicazione alle memorie embedded. Tali studi mostrano l'implementazione di SBST per embedded memory, sia con architetture RISC che con architetture CISC [15, 16].

Una delle ultime innovazioni in questa tipologia di test è l'utilizzazione di soluzioni miste [17, 18], cioè sfruttare il microprocessore per implementare una tecnica del BIST trasparente, così da preservare i dati in memoria. L'algoritmo utilizzato in questo caso è il March Test trasparente [19], riportato in figura 28.

Un march test consiste in una sequenza finita di march elements, un elemento software costituito da una sequenza di operazioni di lettura e/o scrittura applicate a ogni cella della memoria in ordine crescente o decrescente di indirizzi.

Tutte le operazioni di un march element vengono eseguite prima di procedere all'indirizzo successivo. Grazie all'opportuna serie di letture/scritture si identificano i faults delle celle di memoria.

Con riferimento alla figura 28, in cui si riportano il march algorithm (MATS), il march trasperent algorithm (Trasparent MATS) e il simmetric march algorithm (Symmetric Trasparent MATS), si evidenziano i seguenti simboli:

- ↑↓, incremento-decremento indirizzo.
- w, scrittura.
- r, lettura.

In questi algoritmi i valori delle operazioni di lettura e/o scrittura sequenziale dipendono dai dati iniziali, che sono però sconosciuti. Sono pertanto necessari dei passi ausiliari per poter eseguire via software dei test a trasparenza (conservazione dei dati presenti in memoria). A seconda dell'operazione eseguita, può essere necessaria una delle due fasi aggiuntive:

1) Preparazione dei dati nelle operazioni di scrittura, ogni operazione di scrittura utilizza come dati quelli provenienti dalla precedente operazione di lettura o il suo complemento. Di conseguenza, i dati appena letti devono essere complementati prima di essere scritti.
2) Aggiornamento delle maschere dopo le operazioni di lettura: non è possibile confrontare i dati letti con i dati attesi immediatamente dopo la lettura perché i dati attesi in ciascuna operazione non sono noti. Dunque dovrebbe essere generata una opportuna maschera durante il test, che venga aggiornata dopo ogni operazione di lettura e controllata solo alla fine dell'esecuzione di tutto il test.

MATS+	$\{\Uparrow (w_D); \Uparrow (r_D,w_{Dc}); \Downarrow (r_{Dc},w_D)\}$
Transparent MATS+	$\{\Uparrow(r_a); \Downarrow((r_a)c); \Uparrow (r_a,w_ac); \Downarrow (r_ac,w_a)\}$
Symmetric transparent MATS+	$\{\Uparrow (r_a,w_ac); \Downarrow (r_ac,w_a)\}$

Fig. 28: Struttura dei March Test.

Lo studio ha utilizzato quattro diversi March test, mirati ad individuare differenti tipologie di guasto, i risultati mostrano che l'approccio proposto è fattibile e in grado di sostituire HW BIST se necessario, con l'unico svantaggio dell'aumento nel tempo di testing, ma nessuna perdita di copertura dei guasti è stata evidenziata rispetto ai modelli noti.

3 Portable-ATE - Hardware e Software

3.1 Caratteristiche e Funzionalità

Rispetto allo stato dell'arte delle tecniche di testing si è sviluppata una soluzione circuitale innovativa denominata Portable-ATE per il testing dei Memory Test Chip.

Oggi la caratterizzazione, la valutazione delle prestazioni e la rivelazione di eventuali guasti nei chip è effettuata tramite evolute e sofisticate apparecchiature elettroniche denominate Automated Test Equipments (ATEs), capaci di garantire grande parallelismo, velocità e ripetibilità nelle operazioni di test. Ciò è dovuto principalmente al fatto che gli ATE vengono usati per controllare la corrispondenza con le specifiche dei chip che escono dalla divisione di produzione di un industria elettronica e come tali devono testare migliaia di chip nel minor tempo possibile, dato che il tempo di test è una parte non irrilevante del costo globale dei chip stessi. Tali caratteristiche spesso non sono soddisfacenti in fase di ricerca e sviluppo (R&D) dove l'adattabilità e la configurabilità dei test, l'acquisizione dei risultati a seguito di una variazione (anche live) dei parametri o degli stessi test, sono fattori fondamentali nel processo di sviluppo di una tecnologia o di un prodotto.

Per questa ragione è stata sviluppato in collaborazione con STMicroelectronics una scheda di testing a microprocessore denominata Portable-ATE, mostrata in figura 29.

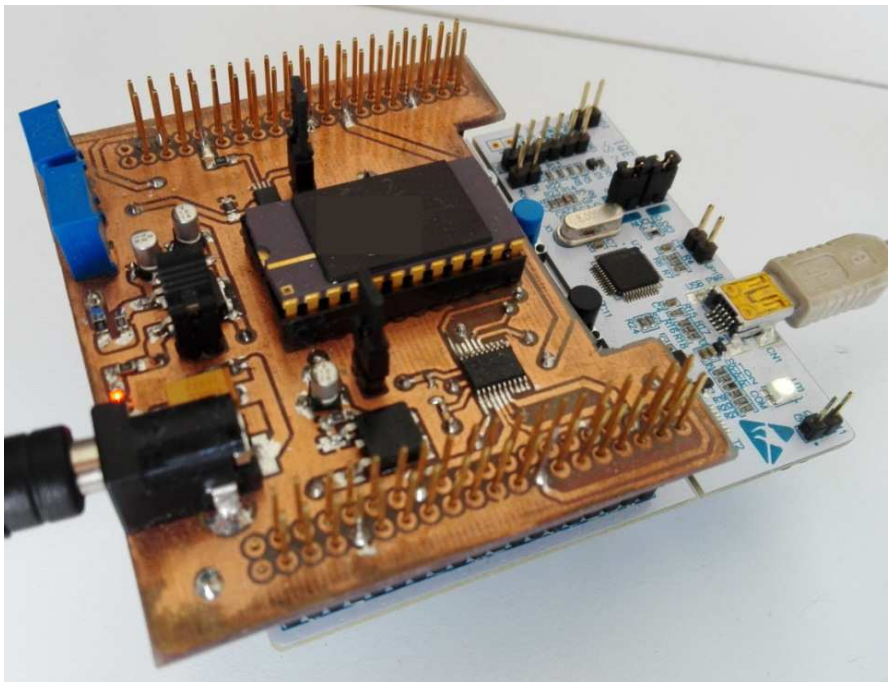


Fig. 29: Portable-ATE con un memory testchip di STMicroelectronics in fase di test.

Questa, con il solo ausilio di un personal computer, è in grado di valutare le performance di un singolo memory testchip con lo stesso grado di affidabilità e riproducibilità di un ATE industriale, anzi aggiungendo caratteristiche di maggiore flessibilità e configurabilità dei test, usualmente necessari in fase di R&D.

Il sistema è basato sulla scheda NUCLEO-F072B di ST Microelectronics, che utilizza un microcontrollore STM32F072RB e su di essa è stata progettata e realizzata una scheda figlia (*daughter board)*, di espansione in grado di ospitare il memory testchip, gestire le alimentazioni, generare ed adattare gli ingressi/uscite necessari al corretto utilizzo della memoria sotto esame. Il microcontrollore è programmato in linguaggio C e comunicando con un personal computer tramite collegamento USB viene facilmente controllato sia utilizzando una Graphic User Interface (GUI), sia inviando direttamente i comandi testuali utilizzando un programma per la comunicazione sulla porta seriale.

In particolare, con il Portable-ATE si possono effettuare letture, scritture e cancellazioni della memoria, test di stress elettrici (Drain, Gate e Bulk Stress), test di ciclatura su zone predeterminate della memoria, sviluppo di algoritmi per l'esecuzione dei processi di scrittura, lettura e cancellazione, debug relativo alle precedenti azioni e tutte le opportune combinazione dei test elencati. Tutti gli algoritmi sviluppati possono essere utilizzati con la massima flessibilità riguardo la scelta dei parametri, quali ad esempio livelli di tensione, tempi di esecuzione, etc.

Questo sistema è stato utilizzato per la prima caratterizzazione dei memory test chip di nuova generazione sviluppati da STMicroelectronics.

3.2 Hardware

3.2.1 Circuito

Lo schema a blocchi del Portable-ATE ed il relativo schema elettrico sono mostrati rispettivamente in figura 30 e figura 31. La scheda di test utilizza un clock di 48MHz, e una scheda di espansione custom, che ospita il dispositivo di memoria in fase di test (MUT – Memory Under Test).Le tensioni di alimentazione necessarie alla memoria per svolgere correttamente le operazioni sono due: una tensione di riferimento di 1.25V (indicata con VDD) e una tensione regolabile (indicata con VCC) che può andare da 3.3V a 6,45V. Queste tensioni sono generate da due regolatori lineari indipendenti e la cui sorgente di alimentazione proviene da un alimentatore plug-in a basso costo (input 220V c.a. – output 12V c.c.).

La tensione di riferimento di 1,25V è stata comunque generata in modo da poter essere regolata a piacimento nel range 0.9÷1.4V, così da rendere la scheda successivamente adattabile ad altri tipi di memoria con il medesimo pattern ma con tensioni di riferimento diverse.

Il valore massimo della tensione VCC è invece pari alla massima tensione consentita ai pad della memoria; in alcuni test infatti è necessario poter aumentare il suo valore. Ciò può accadere ad esempio nel monitoraggio della tensione di bit line durante la programmazione o nell'immissione dall'esterno di un livello adeguato di tensione in fase di test attraverso uno degli analog pads.

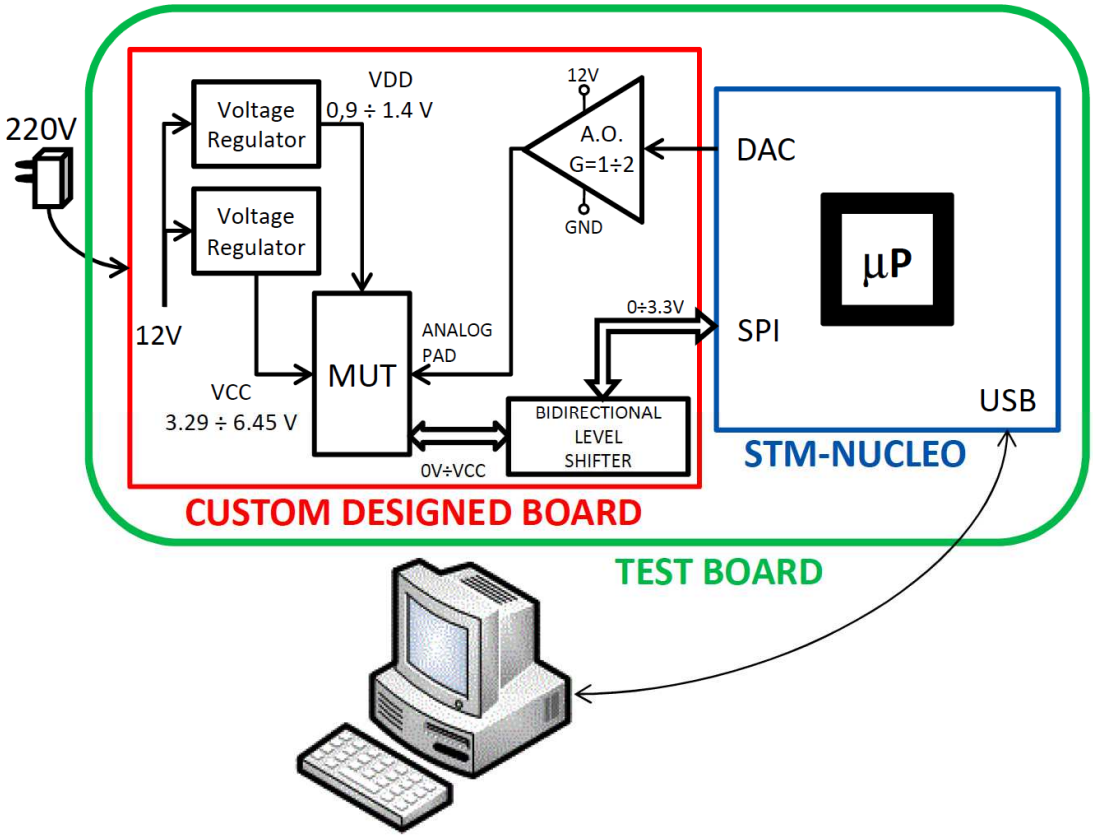


Fig. 30: Schema a Blocchi del Portable-ATE.

Se la VCC deve superare i 3.3V per le esigenze di test della memoria, serve al contempo la necessità di avere un riferimento di 3.3V per poter dialogare con la scheda attraverso il protocollo SPI. Tale protocollo infatti ammette livelli logici 0-3.3V provenienti dalla scheda Nucleo e quindi è stato inserito un level shifter bidirezionale per adattare i livelli di tensione di tutti i segnali necessari, in modo si possa dare alla memoria una tensione più alta senza danneggiare il microcontrollore o avere errori nella comunicazione.

Inoltre, è stata prevista un'uscita analogica controllata dal DAC della scheda Nucleo, la quale opportunamente amplificata, con guadagno regolabile da 1 a 2, è in grado di generare una tensione massima pari a 6,6V. Attraverso opportune impostazioni interne alla memoria, questa tensione può controllare diversi segnali significativi all'interno del MUT quali i segnali di gate, di bitline, di wordline o di bulk. Essa pertanto può essere utilizzata per effettuare diversi tipi analisi quali le distribuzioni dei livelli soglia di tensione, lo stress test e numerose altre indagini.

Qualora risulti necessario, il segnale in questione della memoria può essere scollegato dall'uscita del DAC, tramite jumper, ed essere utilizzato come ingresso per leggere segnali provenienti dalla stessa memoria.

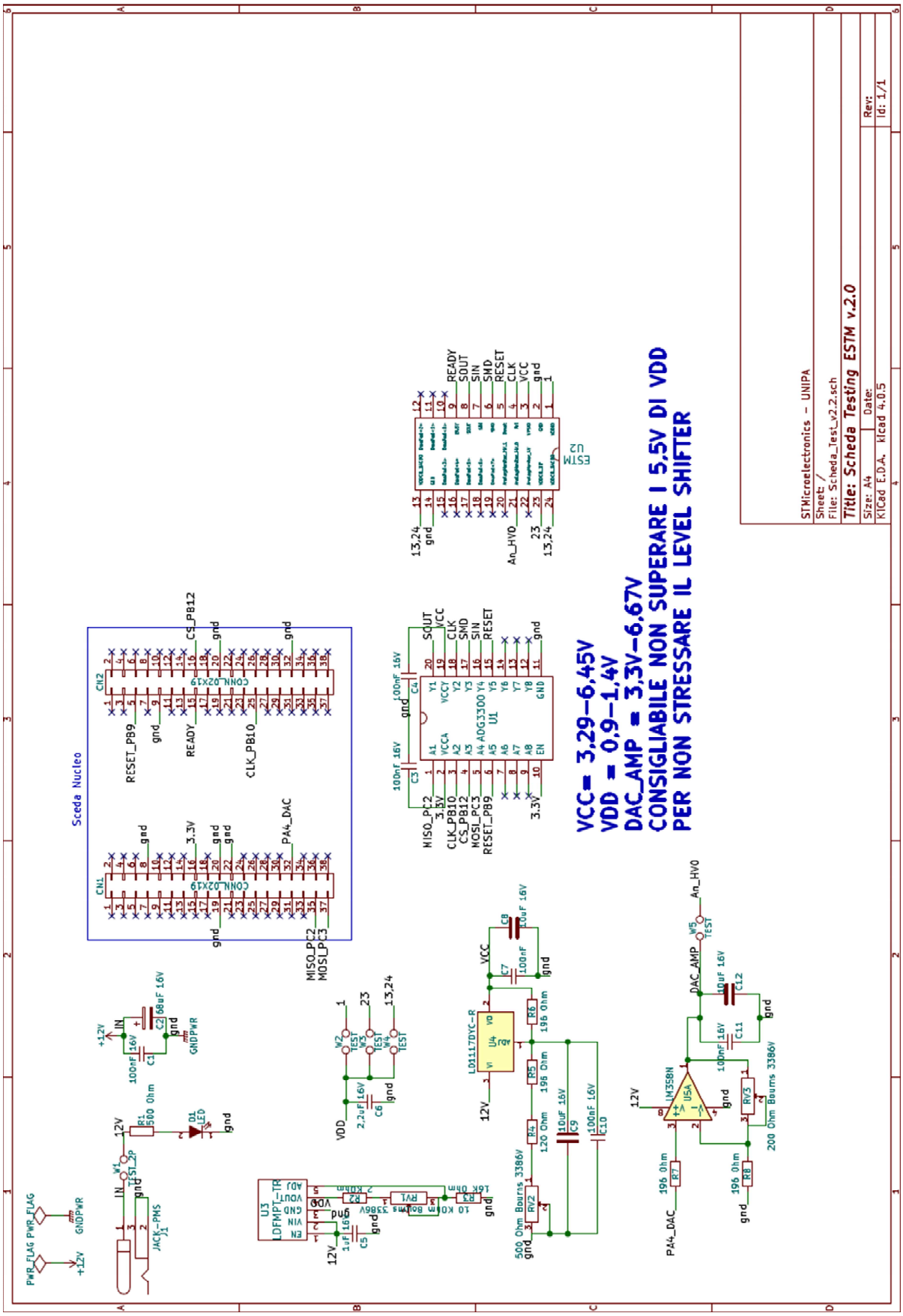


Fig. 31: Schema elettrico della daugther board del Portable-ATE.

3.2.2 Layout

Lo sviluppo del circuito e del layout del Portable ATE è stato effettuato utilizzando il software Kicad, un software EDA gratuito ed altamente flessibile.

Il layout del circuito di figura 31 è stato sviluppato su una scheda doppia faccia facendo attenzione al posizionamento dei connettori e dei componenti in modo da per poter montare la scheda direttamente sulla scheda Nucleo, inoltre è stata fatta una sagomatura così da lasciare accessibili i pulsanti di controllo della Nucleo .

Particolare attenzione è stata posta alle linee di alimentazione ed al piano di massa. I componenti necessari alla regolazione delle tensioni di alimentazione (dei potenziometri lineari multigiro) sono stati posti sul bordo della scheda per una maggiore facilità di accesso. Alcuni jumper sono stati previsti per un debug più veloce ed efficiente, ad esempio un jumper è stato previsto per l'eventuale disconnessione dell'alimentazione in modo da poter eventualmente alimentare la memoria con un alimentatore esterno. Il layout definitivo è mostrato nella figura 32, dove sono mostrati solo i bordi dei piani di massa per una migliore visualizzazione dei componenti e delle piste di collegamento.

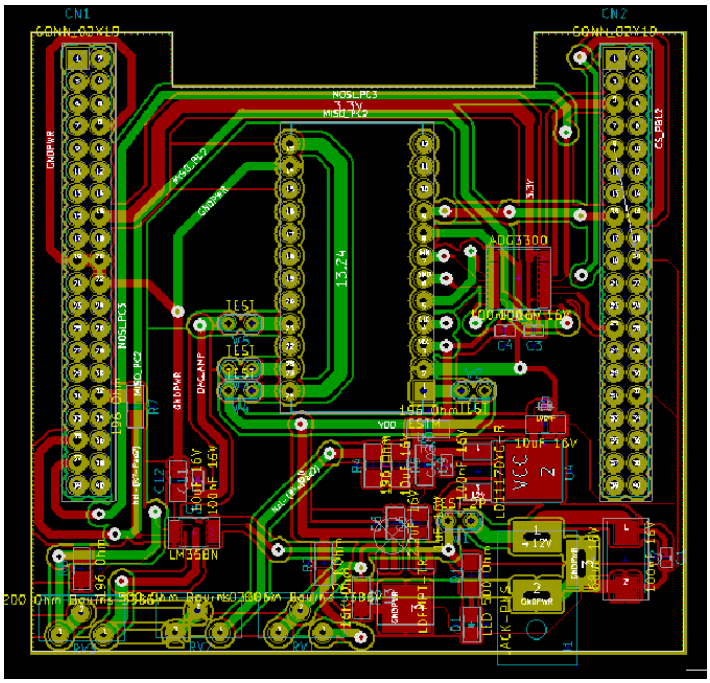


Fig. 32: Layout Portable ATE.

3.3 Software

Il software è stato sviluppato in linguaggio C utilizzando l'ambiente di sviluppo "System Workbench for STM32" e il tool "STM32CubeMX".

3.3.1 Ambiente di Sviluppo

Il firmware del prototipo e della prima versione del Portable ATE è stato sviluppato utilizzano l'ambiente di sviluppo IAR-EWARM, in versione gratuita con limitazione della grandezza del codice a 16Kbyte. Con il progresso del progetto e quindi delle funzioni implementate la limitazione del codice a 16Kbyte è diventata insostenibile, si è dunque scelto di migrare il codice su System Workbench for STM32, programma open source e senza limitazioni. Il System Workbench for STM32 è un ambiente di sviluppo integrato (IDE – Integrated Development Environment), basato su ambiente Eclipse, che integra editor di codici, strumenti di compilazione (compilatore, assembler, linker, etc.) e strumenti di debug; fornisce inoltre la possibilità di poter eseguire il debug sul chip usando l'ST-Link/V2 (integrato nelle schede ST Nucleo). Tutte caratteristiche necessarie per il corretto e completo sviluppo del progetto.

Per l'inizializzazione del codice è stato utilizzato il tool STM32CubeMX che, grazie ad un'interfaccia grafica semplice e intuitiva, permette la configurazione iniziale delle periferiche interne del microcontrollore generando anche il codice C corrispondente alle inizializzazioni delle periferiche e delle interfacce necessarie per il progetto. Infatti, tramite interfaccia grafica una volta selezionata la scheda usata o il microcontrollore STM32 corrispondente si possono configurare tutti i pin di I/O necessari, eventualmente attivando le periferiche interne corrispondenti; si possono settare i clock interni del microcontrollore con la possibilità di scegliere tutti i parametri con valori compatibili a quelli consentiti dal microcontrollore; si può stimare il consumo del microcontrollore; si possono configurare le modalità di funzionamento delle periferiche (GPIO, USART, etc.) settando i parametri desiderati e se necessario anche dei componenti software aggiuntivi (RTOS, USB, TCP/IP, etc.).

Quando la configurazione è stata completata, si genera il corrispondente codice C di inizializzazione, che può essere personalizzato per diversi ambienti di sviluppo (Ewarm, System Workbench, etc.) senza la necessità di apportare modifiche alla configurazione del microcontrollore.

Lo stesso progetto può essere successivamente modificato e la nuova generazione del codice non modifica il codice utente eventualmente aggiunto.

3.3.2 Funzioni

La memoria comunica con la scheda Nucleo tramite un protocollo SPI custom, la frequenza di funzionamento è pari a 12MHz, la massima supportata dai chip di memoria usati per i test. Le funzioni elementari implementate sono cinque:

- Scrittura dei registri interni;
- Lettura dei registri interni;
- Scrittura dell'array;
- Lettura dell'array;
- Lettura multipla dell'array;

Tramite il settaggio dei registri interni ed utilizzando la lettura e scrittura dell'array si possono effettuare molteplici operazioni di test, in particolare si sono sviluppate, generando il codice personalizzato, le seguenti routine:

- Reset della memoria;
- Ciclo di power-on, in cui si settano tutti i registri interni con i valori di default preimpostati;
- Lettura e salvataggio di tutti i registri interni;
- Lettura o scrittura di un singolo registro,
- Lettura o programmazione di una word in array;
- Lettura tramite BIST interno del checkerboard sulle righe desiderate;
- Lettura tramite BIST interno del checkerboard inverso sulle righe desiderate;
- Lettura del numero di righe desiderato con ECC attivo;
- Lettura del numero di righe desiderato con ECC spento;
- Programmazione di un numero di righe impostato dall'utente con il valore inserito;
- Programmazione tramite BIST interno di un checkerboard sulle righe desiderate;

- Programmazione tramite BIST interno di un checkerboard inverso sulle righe desiderate;
- Soft programmazione tramite BIST interno delle righe desiderate;
- Cancellazione di una riga (pagina);
- Cancellazione del numero di righe desiderato;
- Refresh tramite BIST interno delle righe desiderate;
- Routine che effettua la distribuzione del numero di righe e nell'intervallo di tensione (simmetrico) impostato dall'utente;
- Routine per la valutazione del Gate Stress;
- Routine per la valutazione del Drain Stress;
- Varie routine di ciclatura, dalle statiche (valori preimpostati e non adattabili) alle dinamiche con verify e differenti tipi di adattamento al fail (incremento delle tensioni di program e/o di erase) ed anche con refresh dei valori memorizzati per emulare completamente lo user-mode;
- Routine per la calibrazione del DAC;
- Diverse routine per l'emulazione degli algoritmi di program/erase con verify ed il relativo debug;
- Routine per l'emulazione dell'algoritmo di Refresh;

Con l'implementazione di queste routine è stato possibile effettuare la caratterizzazione ed il debug di una nuova tipologia di memoria, a partire dalla caratterizzazione elettrica della singola cella, al comportamento dell'array, alla valutazione degli stress elettrici subiti dalla cella nel normale funzionamento (in particolare analisi di gate e drain stress), alla valutazione del tempo di vita della memoria e dei possibili accorgimenti per aumentare il tempo di vita stesso. In particolare si è implementato l'algoritmo di program con verify e si è valutato l'impatto di differenti algoritmi di erase con verify.

Lo studio si è concentrato sull'algoritmo di erase verify dato che, come riportato in letteratura [20,21] e come dimostrato da dati sperimentali, la chiusura della finestra di erase, e la conseguente inutilizzabilità della memoria, è molto più pronunciata di quella di program e dunque quella che causa maggior deperimento delle prestazioni della memoria.

Nel prossimo capitolo si analizzeranno i risultati delle analisi sugli stress elettrici, degli algoritmi di ciclatura e dell'implementazione degli algoritmi di gestione della memoria, program ed erase con verify e refresh.

4 Analisi degli Stress e Sviluppo degli Algoritmi

In questo capitolo saranno descritte le analisi eseguite con il Portable ATE su memory test chip di STMicroelectronics con tecnologia a 40nm, rispettivamente di 1 MB e di 512KB, le cui word sono formate da 32 bit di dato più 6 bit di ECC. Le memorie utilizzate sono programmate per Channel Hot Electron Injection (CHEI) e cancellate per Fowler-Nordheim Tunneling.

4.1 Analisi di Drain Stress

Il fenomeno del drain stress è di rilevante importanza durante la fase di programmazione, gli effetti della programmazione e della cancellazione sono stati descritti nel paragrafo 2.2.

I circuiti per valutare il drain stress vengono implementati con tecniche tradizionali per forzare tutte le word line a massa, tutti i drain ad alta tensione (tipicamente intorno a 5V) e i source flottanti per un tempo dipendente da quanti cicli di Program/Erase si vogliono emulare. L'applicazione dell'alto valore di tensione può essere eseguita o utilizzando le pompe interne o fornendo tale tensione dall'esterno, una volta abilitati i registri interni che permettono tale modalità di funzionamento della memoria. Un grafico di una tipica analisi di drain stress per le celle programmate, riportato in figura 33, mostra l'effetto sulle distribuzioni di soglia di diversi valori tensione di drain al variare del tempo di applicazione.

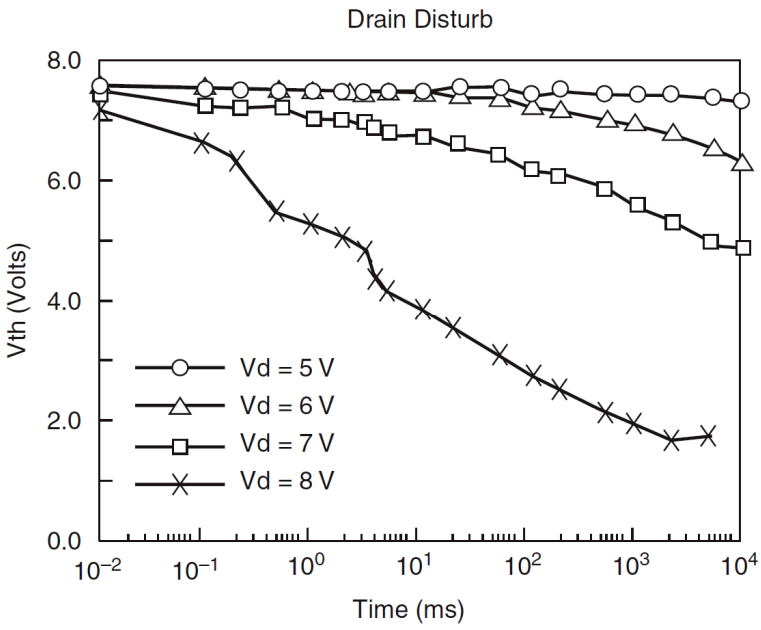


Fig. 33: Effetto del drain stress, dovuto a FN Tunnelling, sulle distribuzioni delle tensioni di soglia delle celle programmate. (Figura 7(a) di [10])

Le analisi di stress effettuate non sono state fatte utilizzando il metodo descritto precedentemente, ma si è voluto effettivamente verificare l'effetto di continui cicli di Program/Erase sulle celle non interessate da queste operazioni. Per rendere l'analisi ancora più significativa abbiamo condotto dei test comparativi all'interno dello stesso chip di memoria, utilizzando array di celle "invecchiate" e di celle "giovani"; così da poter osservare gli effetti che lo stress applicato ha sui due semi-array sottoposti alle medesime prove. Ci si attendono qualitativamente effetti di deterioramento delle prestazioni più marcati nelle celle invecchiate, a causa del degrado accumulato nell'ossido. In particolare, il test è stato eseguito su un semi-array (256KB), in cui è stata eseguita una prima fase di invecchiamento composta da 128000 cicli di program/erase a tensione costante con valori applicati riportati in tabella 2.

Tabella 2: Tensioni applicate per la fase di invecchiamento del semi-array prima del test sul drain stress.

	Tensione di Gate (V_G)	Tensione di Drain (V_D)	Tensione di Bulk (V_B)	Tensione di Source (V_S)
Program	10 V	5V	GND	GND
Erase	-10V	Floating	10V	Floating

Su questo semi-array invecchiato è stato scritto, tramite BIST interno, una checker-board e successivamente sul semy-array restante sono stati eseguiti un totale di 100000 cicli di program/erase, raggruppati in step di 20000 cicli. Alla fine di ogni step da 20000 cicli sono state estratte le distribuzioni dell'andamento delle soglie della checker-board ed i risultati sono mostrati nella figura 34. L'asse X riporta le tensioni di soglia (ΔV_{TH}) relative al valore di sensing, mentre l'asse Y riporta il numero di celle in cui occorre un particolare valore di soglia leggibile dalla corrispondente ascissa. Gli andamenti, raccolti sperimentalmente, evidenziano una convergenza prevista delle distribuzioni delle celle programmate e cancellate verso il valore centrale, segno tangibile dell'accumulo di stress che deteriora i valori di soglia delle celle.

Per convalidare i risultati ottenuti con il Portable-ATE lo stesso test è stato eseguito su un ATE classico (modello Teradyne J750) in dotazione al centro di Ricerca e Sviluppo di ST Microelectronics di Rousset (Francia). In particolare, il test è stato sviluppato su un testchip con architettura identica e proveniente dello stesso lotto di produzione di quello di Palermo ed eseguendo un pre-invecchiamento attraverso 100000 cicli di

program/erase, un successivo test di checker-board standard ed infine sono stati eseguiti 100000 cicli di program/erase, raggruppati a passo variabile, ed i cui risultati sono mostrati in Figura 35.

Si nota che i grafici 34 e 35 mostrano lo stesso andamento delle curve all'aumentare del numero di cicli, in particolare si evidenzia in entrambi i casi l'attesa chiusura delle finestre delle celle cancellate e delle celle programmate. Inoltre, gli intervalli di tensione e gli shift nei due casi sono i medesimi, si nota anche che la forma a campana delle celle programmate tende ad essere leggermente più stretta e con un valore di picco più alto in entrambi i casi. Con i risultati ottenuto da questo ed altri test, effettuati in vari momenti del percorso di dotttorato, è stata effettuata la validazione del Portable ATE rispetto al convenzionale ATE utilizzato da STMicroelectronics per la validazione delle tecnologie e dei prodotti.

Oltre alle analisi di questo tipo sono state effettuate ulteriori analisi di drain stress immettendo all'interno delle fasi di test una fase di cottura in forno (bake), vista l'importanza che tale step ha per la validazione dei componenti, così come già descritto nel capitolo 2.

In pratica su alcune celle già invecchiate attraverso 100000 cicli di Program/Erase, come descritto in precedenza, è stata effettuata una fase di bake a 150°C per 24 ore. La fase di bake è stata effettuata per verificare la possibilità che una parte dello shift delle caratteristiche fosse dovuto al fenomeno di trapping dell'ossido.

Dopo il bake, è stata letta la checker-board che era stata scritta in precedenza per controllare l'eventuale presenza di variazioni del contenuto delle celle dovute al solo riscaldamento.

In un secondo tempo si è effettuata la valutazione del drain stress tramite una seconda ciclatura del restante array, con step di raccolta delle distribuzioni pari a 40000, 70000 e 100000 cicli. I risultati di tali prove sono mostrati in figura 36, essi evidenziano un effettivo de-trapping all'interno dell'ossido, ed una successiva diminuzione dello shift dovuto al drain stress sulle distribuzioni. Sono stati effettuati ulteriori test al variare della tensione di bitline (±0,5V) e della corrente di program, questi test sono serviti per comprendere meglio e verificare il funzionamento della memoria a lungo termine e dunque per la scelta dei parametri di funzionamento standard al fine di ottimizzare il life-time e la reliability della memoria.

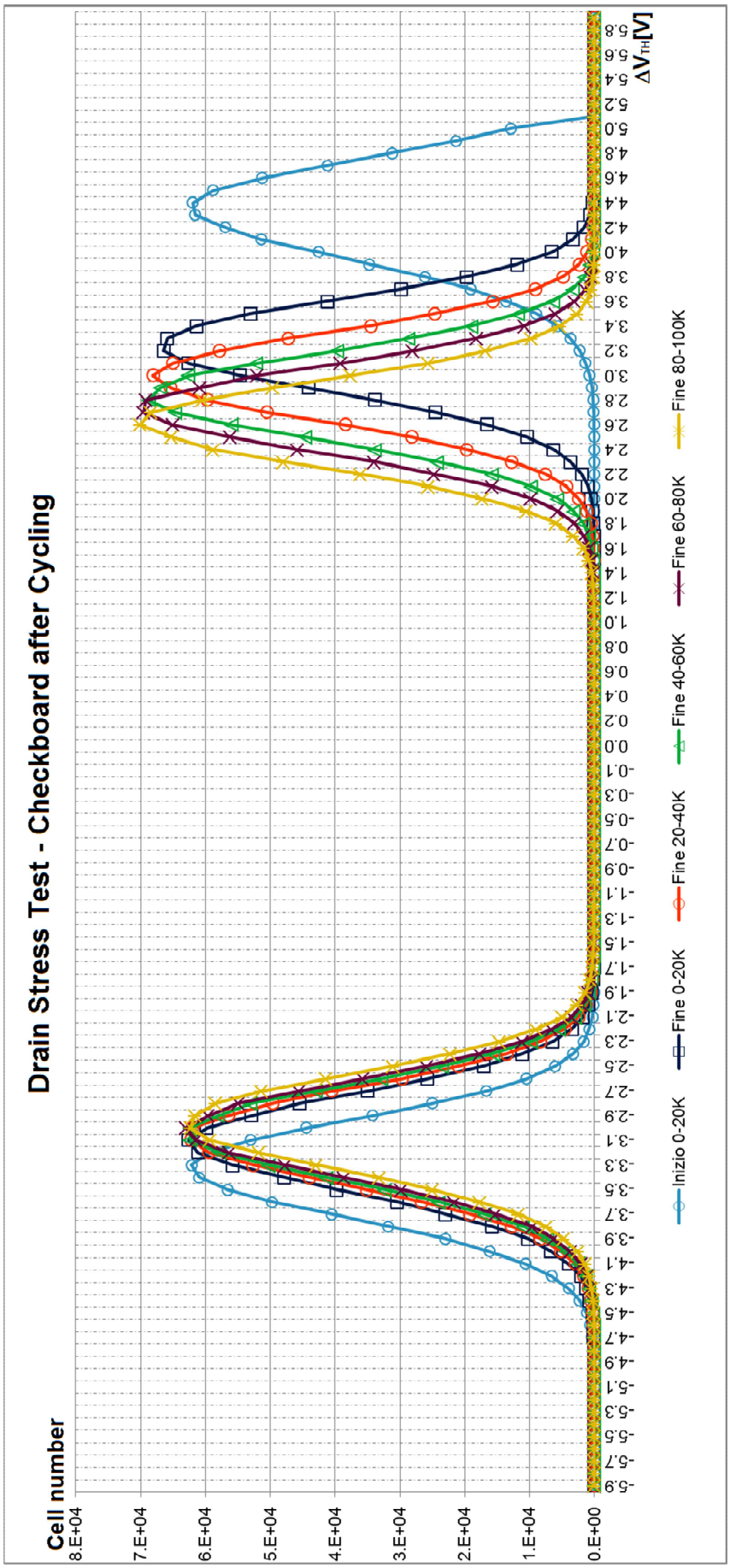


Fig. 34: Risultati dell'analisi di drain stress effettuata con Portable ATE su celle precedentemente invecchiate.

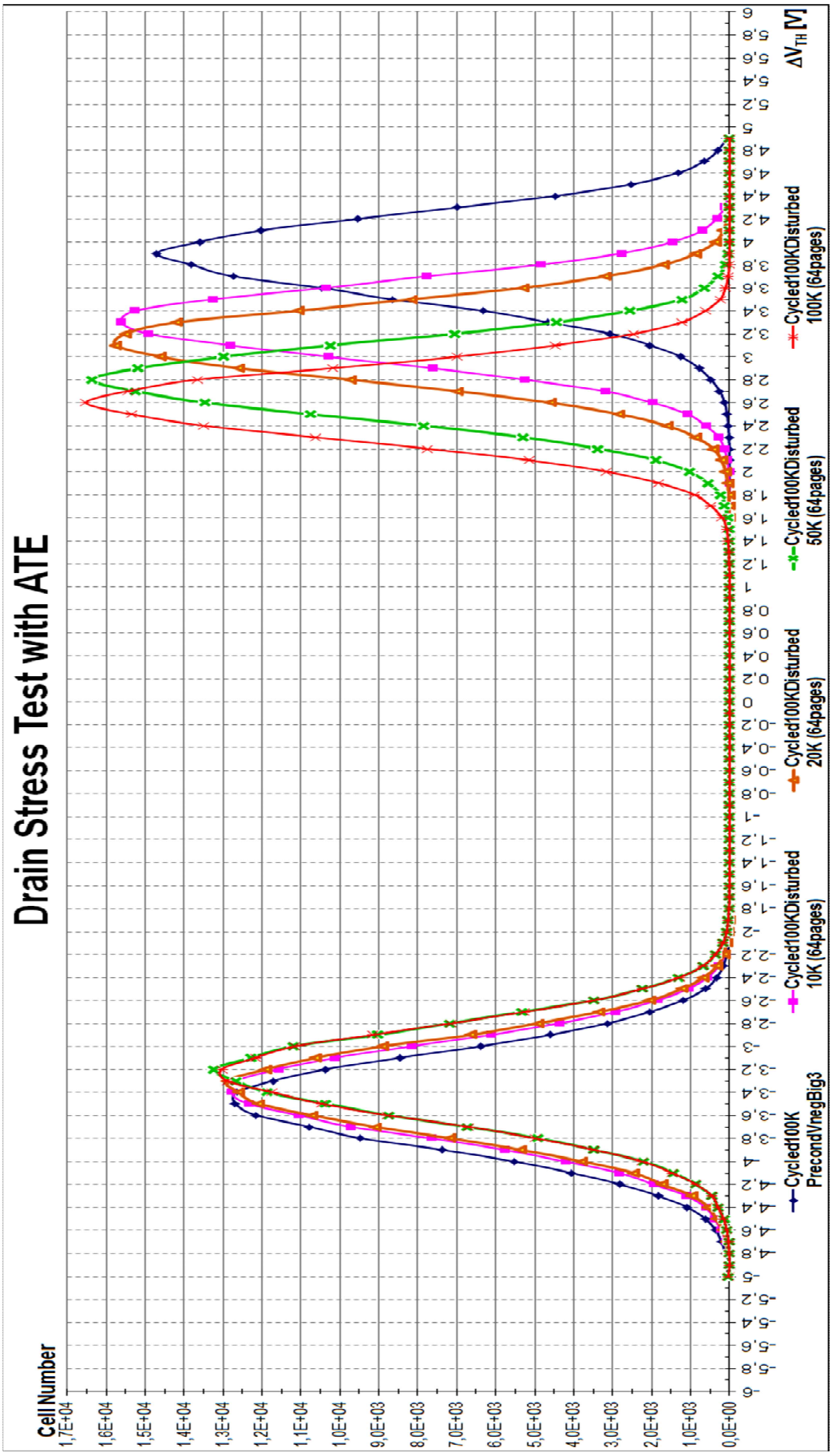


Fig. 35: Risultati dell'analisi di drain stress effettuata con Teradyne J750 su celle precedentemente invecchiate.

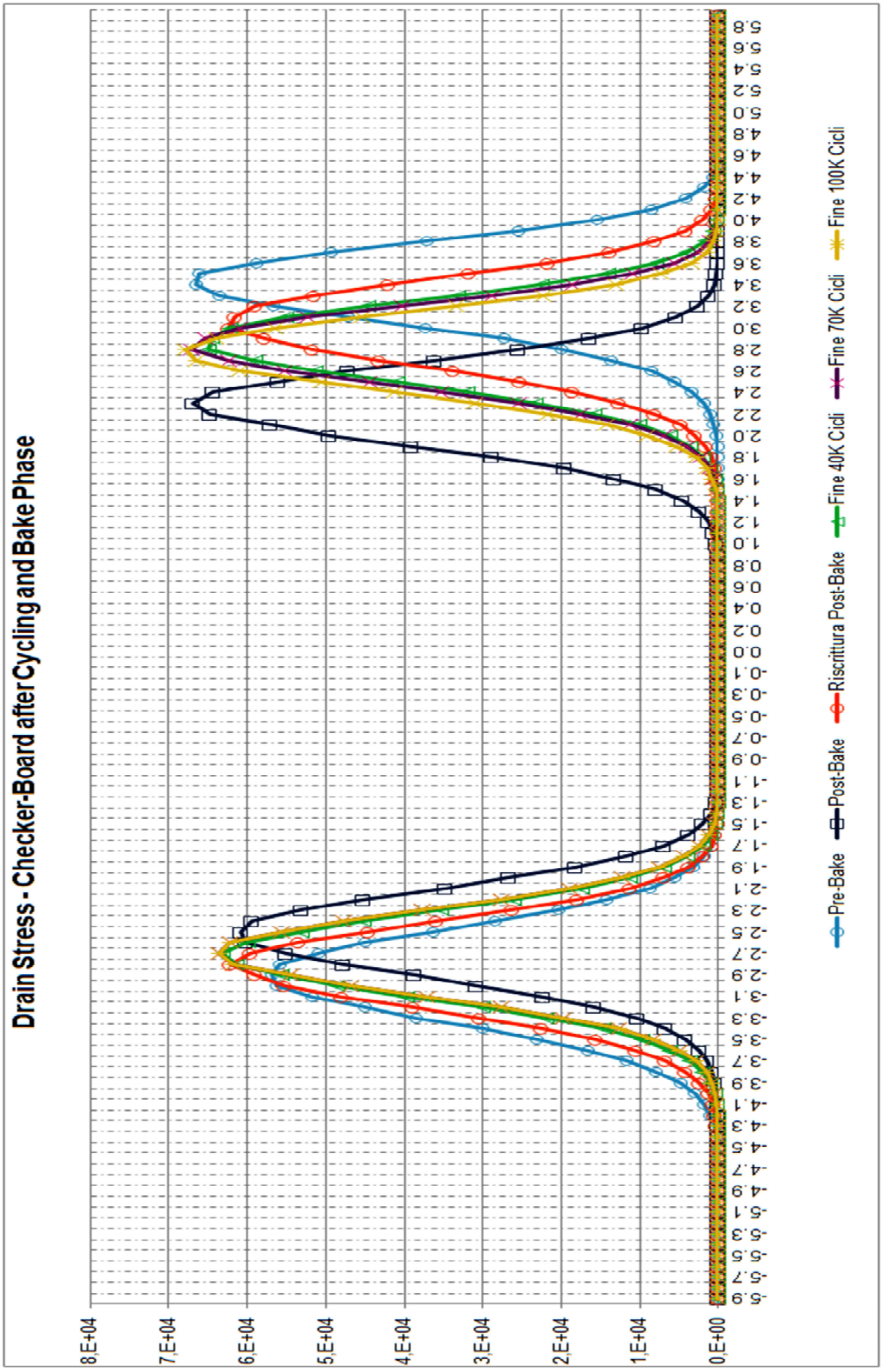


Fig. 36: Risultati dell'analisi di drain stress effettuata con Portable ATE su celle precedentemente invecchiate e con fase di bake intermedia.

4.2 Analisi di Gate Stress

L'analisi di gate stress non è stata la classica analisi per la valutazione del disturbo di program, descritto nel paragrafo 2.2; essa infatti prevede l'applicazione di un'alta tensione (tipicamente intorno a ±10V) di word line, tutti i drain ed i source a massa per un tempo dipendente da quanti cicli di Program/Erase si vogliono emulare. I tipici grafici risultanti da questo tipo di analisi sono riportati in figura 37; la figura a) mostra il gate stress delle celle programmate dovuto al leakage poly-poly, la figura b) mostra il gate stress delle celle cancellate dovuto al leakage dell'ossido di tunnel.

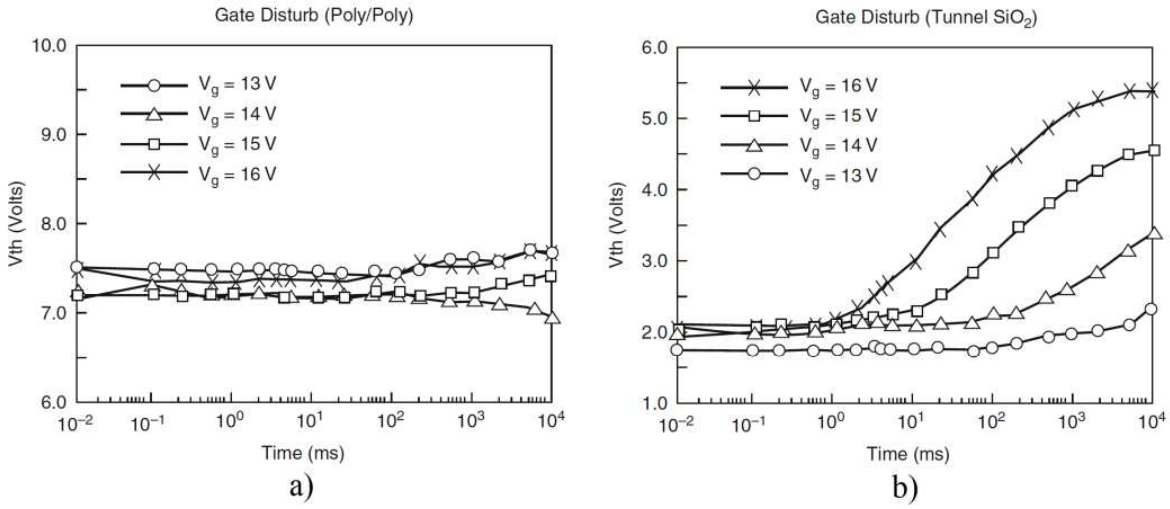


Fig. 37: Gate stress in fase di program. a) celle programmate b) celle cancellate

Nei dispositivi sotto esame si è voluta fare un'analisi di gate stress su celle invecchiate per valutare il possibile errore che interviene nel caso di estrazione di successive distribuzioni dalla medesima riga dell'array sotto test. Infatti, quando viene effettuata una generica estrazione delle distribuzioni di soglia, viene applicata al gate una rampa di valori quantizzata di tensione, fino al raggiungimento di un valore finale intorno a 15V. Se si effettuano estrazioni successive di distribuzioni di soglia dalla medesima riga, il risultato della distribuzione corrente può essere influenzato dalle precedenti e dunque ottenere risultati non corretti, in particolare questo effetto è marcato per le distribuzioni delle celle cancellate, come riportano anche i risultati classici del gate stress riportati in figura 37.

L'analisi è stata svolta su celle di memoria precedentemente invecchiate con 100000 cicli di program/erase con le tensioni riportate in tabella 2, è stato dunque scritto una checker-board utilizzando il BIST interno alla memoria.

Successivamente, sono state effettuate le estrazioni delle distribuzioni di soglia, che in questo caso consiste in una procedura di lettura dell'array sotto test con tensione di gate imposta esternamente e successiva raccolta delle informazioni relative allo stato delle celle, conteggiandone i valori risultanti 1 e 0. Sono state effettuate estrazioni successive delle distribuzioni di soglia applicando una rampa di tensione di gate con escursione di 12V. In figura 38 sono mostrate le distribuzioni delle celle cancellate che, come atteso, subiscono uno shift verso destra, essendo tale è più marcato nelle prime distribuzioni e via via più lieve mano mano che si procede con le letture.

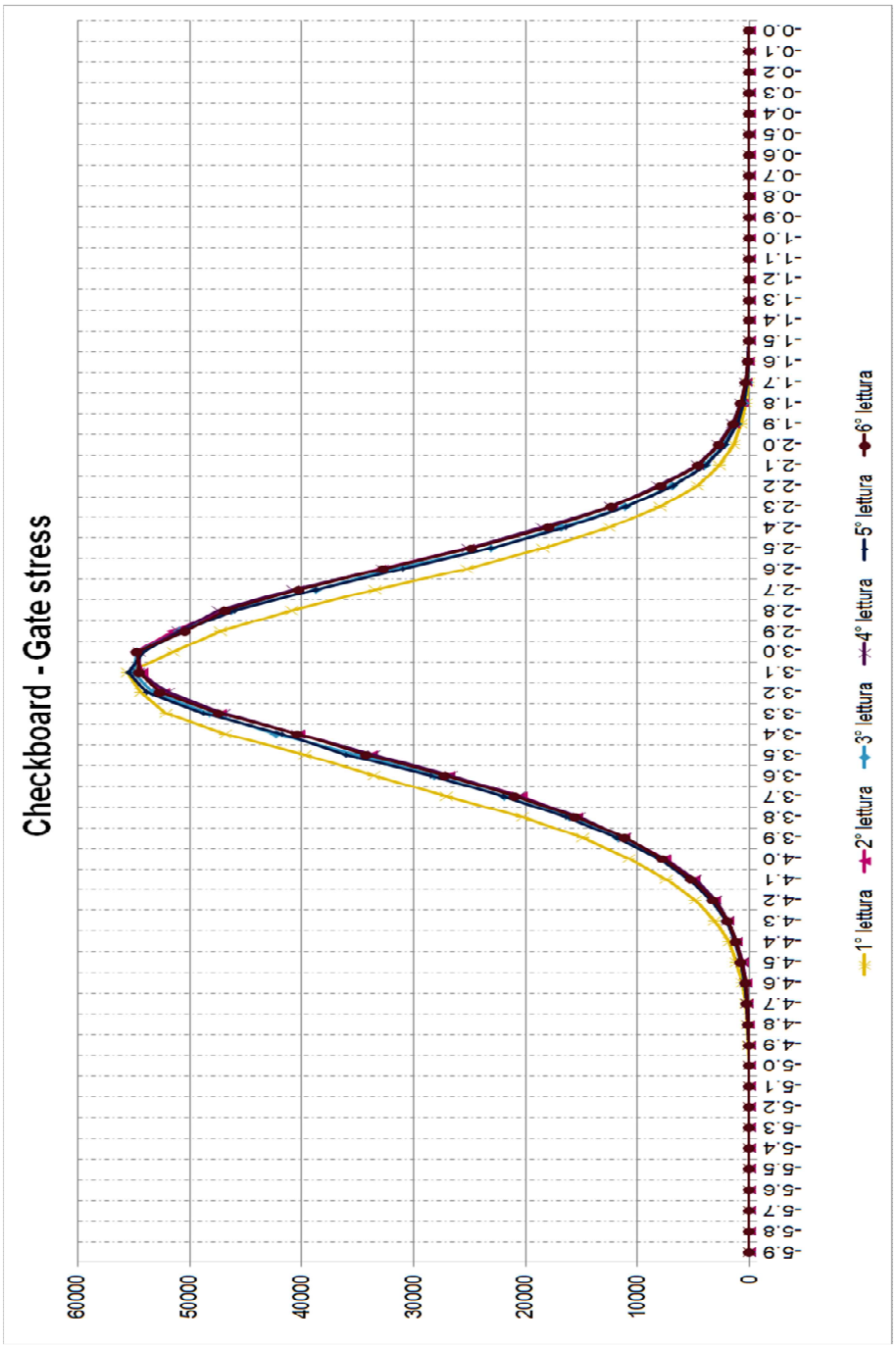


Fig. 38: Gate Stress dovuto a letture successive per l'estrazione delle distribuzioni di soglia delle celle cancellate.

Il movimento delle distribuzioni evidenziato è stato studiato in letteratura ed il meccanismo, denominato effetto SILC (Stress Induced Tunnel Oxide Leakage current), occorre quando le celle vengono sottoposte, come nel caso in esame a stress di tensione di tipo bipolari (±10V) [21]. Come mostrato nella figura 39, la corrente di dispersione manifesta due regioni, una regione di decadimento (Decay Region) e una regione stazionaria (Steady State Region). All'inizio dello stress di lettura, dopo i cicli di program/erase, il ΔVth/Δt decade rapidamente, questo è causato sia dal rapido decadimento dovuto al SILC sia dal decadimento del numero di portatori intrappolati nell'ossido di tunnel. Dopo questa prima fase si raggiunge una regione di stato stazionario in cui ΔVth/Δt diminuisce gradualmente all'aumentare del tempo del disturbo di lettura.

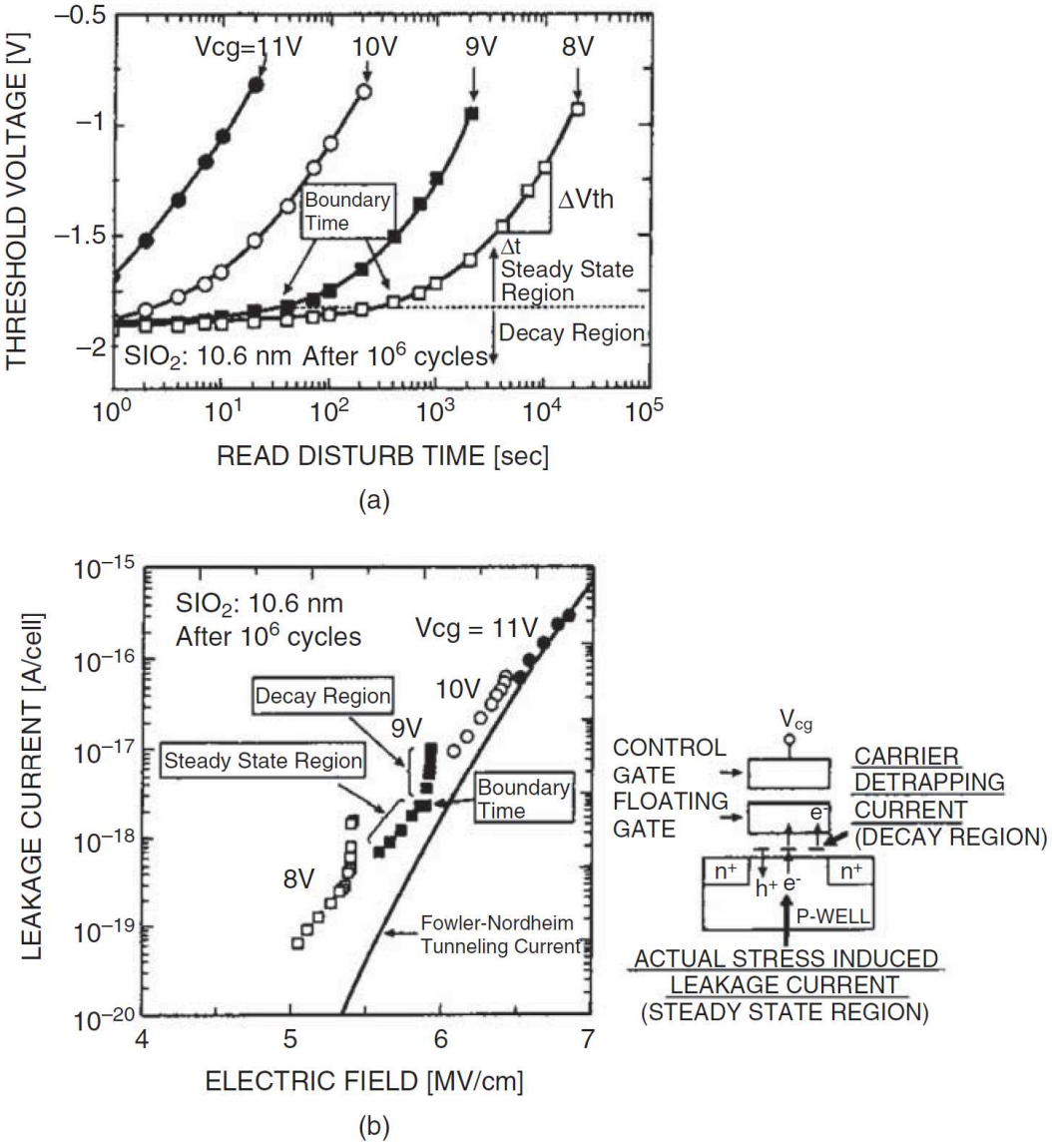


Fig. 39: Caratterizzazione del fenomeno del SILC basato sullo shift delle tensioni di soglia di Flash Memory sotto condizioni di disturbi di lettura. (figura 2 di [11])

4.3 Analisi di Ciclatura

In questo paragrafo vengono riportati gli algoritmi di ciclatura che sono stati utilizzati per effettuare varie tipologie di test, ad esempio il drain/gate stress descritti nei paragrafi precedenti sono stati svilutppati utilizzando ciclature di tipo standard, mentre per la valutazione dei tempi di vita sono state utilizzate ciclature standard con verify e ciclature dinamiche con verify.

In particolare durante le fasi di test sono stati sviluppati tre algoritmi di ciclatura. Il primo algoritmo, denominato "ciclatura standard", il cui diagramma di flusso è mostrato in Figura 40, è una ciclatura a tensione di programmazione e cancellazione prefissata senza l'introduzione della fase di verify ed in cui sono settabili dall'utente i seguenti parametri:

- Il numero di cicli da effettuare, memorizzati nella variabile N;
- La riga di inizio e il numero di righe su cui effettuare la ciclatura, memorizzati in due variabili (denominate rispettivamente Address e Row_Number);
- I valori di tensione a cui si vuole effettuare il test. In particolare nella fase di cancellazione viene settata la tensione di bulk all'interno di un campo di valori variabile in modo quantizzato da 6,52V a 11,27V e memorizzata nella variabile $V_{ER;}$ mentre in fase di programmazione la tensione di gate (settabile sempre nel range 6,52V÷11,27V) viene memorizzata nella variabile V_{PR}.

Le operazioni di programmazione e di cancellazione sono state effettuate utilizzando il BIST interno del memory test chip, così da non dover implementare via software la gestione degli indirizzi.

Questa algoritmo è stato utilizzato sia per valutare l'invecchiamento delle celle su cui si volevano effettuare ulteriori analisi, come quelle riportate nel paragrafo 4.1 e 4.2, sia per la valutazione della degradazione delle caratteristiche delle distribuzioni di soglia delle celle programmate e delle celle cancellate, al variare di alcuni parametri.

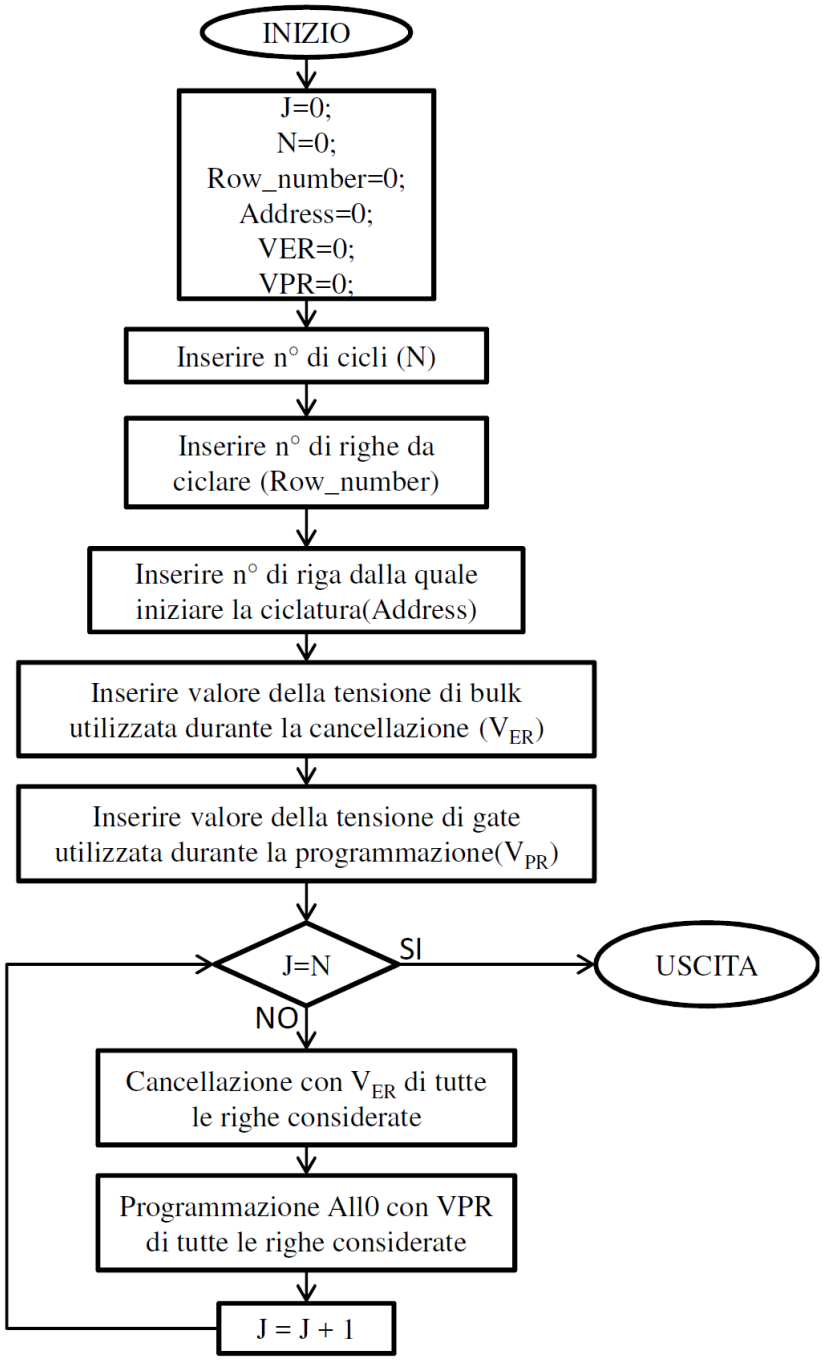


Fig. 40: Diagramma di Flusso dell'algoritmo di ciclatura standard.

Il secondo ed il terzo algoritmo di ciclatura sono degli algoritmi di "ciclatura completi", nel senso che emulano il funzionamento della memoria in normali condizioni operative e con tutte le procedure necessarie implementate quali in particolare:

- Programmazione con verify,
- Cancellazione con verify
- Refresh;

Queste tre procedure verranno descritte in maniera puntuale nel paragrafo seguente. La differenza fra i due algoritmi consiste invece nel tipo di programmazione effettuata, nel secondo algoritmo viene svolta una programmazione "all zero" (All0), mentre nel terzo viene effettuata una programmazione di tipo checkerboard; in quest'ultimo caso quindi, in base alla programmazione, cambierà conseguentemente la maschera da utilizzare per il verify.

Analizziamo adesso il flusso dell'algoritmo di ciclatura completo, mostrato in figura 41, e come esso viene eseguito all'interno dell'array.

L'algoritmo svolge due operazioni principali, cicla la parte di memoria desiderata (operazione di cancellazione e programmazione con verify per ogni riga) ed effettua, sempre per riga, il refresh dell'array; le due operazioni sono fra loro disgiunte e indipendenti. Nelle memorie testate in questi esperimenti ogni riga è formata da 128 word ed ogni word è a sua volta costituita da 32 bits per il dato e 6 bits per l'ECC.

L'operazione di ciclatura (cancellazione e programmazione con verify) viene effettuata nel semy-array selezionato, superiore o inferiore, a partire dalla riga selezionata per il numero di cicli (N) e il numero di righe (Row_Number) desiderati.

L'operazione di refresh avverrà un numero di volte pari al numero di cicli, iniziando dall'indirizzo memorizzato nelle prime 16 word dell'ultima riga del semi-array selezionato (Row_Counter), come verrà meglio descritto nel paragrafo 4.4.3.

Come riportato in figura 41, l'algoritmo in fase di avvio chiede all'utente alcuni valori per i parametri d'interesse:

- Bench Superiore o Inferiore (0 o 1). Definisce in quale parte della memoria (bench) da 512KB si effettueranno le operazioni. (Ogni bench da 256KB è formato da 512 righe. Il bench superiore è definito dagli indirizzi 00000H÷0FFFFH, mentre il bench inferiore dagli indirizzi 10000H÷1FFFFH).
- Numero di cicli (N). Definisce il numero di cicli (erase+program) che si vogliono effettuare.
- Row_Number. Definisce il numero di righe che si desidera ciclare (dato che ogni bench è formato da 512 righe questo parametro può variare da 0 a 511, perché l'ultima riga di ogni bench è utilizzata come contatore di riga per le operazioni di refresh).

- V_{PR}, V_{ER}: definiscono le tensioni applicate rispettivamente al control gate e al bulk in fase di programmazione e cancellazione.
- V_{EV}, V_{RV}, V_{VP} ($V_{EV}<V_{RV}<V_{PV}$): definiscono le tensioni di control gate applicate in fase di lettura per la verifica (verify) nelle operazioni rispettivamente di programmazione (V_{VP}), cancellazione (V_{EV}) e lettura (V_{RV}).

Decisi opportunamente tali parametri, l'algoritmo inizia identificando l'indirizzo della riga di memoria in cui effettuare le operazioni di program ed erase in base al semi-array e al numero di riga selezionate dall'utente.

Ad ogni ciclo, per il numero di righe stabilito (Row_Number), verranno eseguite le tre sub-routine:

- Refresh: passando le variabili: Array e Vvr; per identificare il semi-array su cui si sta operando e la tensione con cui bisogna effettuare il verify;
- Cancellazione con verify passando le variabili: Indirizzo, V_{ER} e V_{EV}; per identificare l'indirizzo su cui si deve effettuare l'operazione, la tensione di cancellazione e la tensione di erase verify.
- Programmazione (All0 o checkerboard) con verify: passando le variabili: Indirizzo, V_{PR} e V_{PV}; per identificare l'indirizzo su cui si deve effettuare l'operazione, la tensione di programmazione e la tensione di program verify;

Completate le sub-routine si incrementa l'indirizzo di riga per passare alla successiva, fino al completamento di tale ciclo per il numero di pagine stabilito. Questo viene poi ripetuto fino al completamento del numero di cicli settato (N).

Questa tipologia di algoritmo è stata utilizzata per valutare il tempo di vita della memoria al variare delle tipologie di stress applicati, valori assoluti di tensione, tipologia degli incrementi in caso di non verifica dell'informazione e durata delle operazioni, come verrà descritto nel paragrafo 4.5.

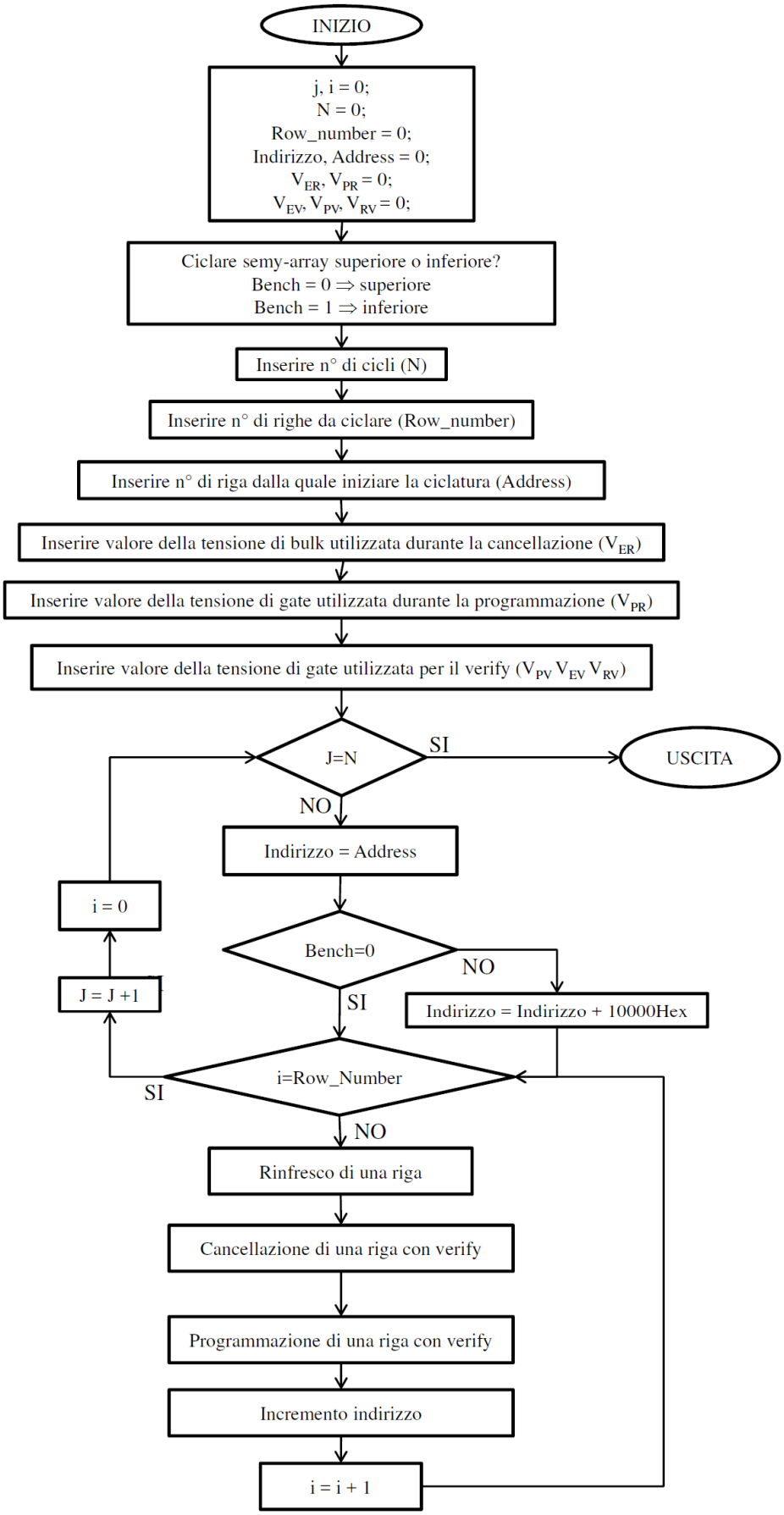


Fig. 41: Diagramma di Flusso degli algoritmi di ciclatura completa.

4.4 Sviluppo Algoritmi di Gestione

Gli algoritmi di gestione della memoria sono stati sviluppati e testati all'interno della memoria o facendo uso del loro microprocessore interno o sfruttando la presenza del microprocessore del Portable ATE che ha permesso la completa emulazione del PEC (Program Erase Controller). Normalmente l'implementazione di questi algoritmi è svolta con programmi di sviluppo e simulazione digitali; l'innovazione del Portable ATE è stata quella di poter sviluppare gli algoritmi e poterli testare direttamente in memoria, prevedendo una fase di ottimizzazione derivante da misure fatte direttamente sulla memoria (in field) anziché simulate al computer.

4.4.1 Program All 0 with verify

Nella figura 42 è riportato il diagramma di flusso della sub-routine "program all 0 with verify" che necessita delle seguenti variabili provenienti dalla routine principale:

- Address, identifica l'indirizzo di riga su cui deve essere effettuata l'operazione.
- V_{PR}, imposta la tensione di control gate (6,52V÷11,27V) in programmazione.
- $V_{PV,}$ imposta la tensione di control gate nella lettura per il program verify. Tale tensione è maggiore del valore nominale di lettura di una quantità pari al margine (solitamente fra 1÷2V) che si vuole avere rispetto alla coda del programmato.

Dopo aver caricato le variabili dalla routine principale, l'algoritmo svolgerà l'operazione di program all zero per l'intera riga selezionata. Successivamente per un numero di cicli pari a 128, pari al numero di word che compongono la riga, si effettuerà l'operazione di program verify.

Il program verify è composto dalle seguenti operazioni:

- Lettura della word di indirizzo Address+i con tensione di control gate di verify program V_{PV}. La tensione V_{PV} (di solito $V_{PV}\geq 1V$), impostata dall'utente nella fase iniziale della routine di ciclatura, è normalmente maggiore della tensione di read verify (V_{RV}) in modo da dare un margine di sicurezza alla distribuzione delle celle programmate rispetto alle condizioni standard di lettura.
- Verificare se il contenuto letto è uguale a zero (all 0), cioè si controlla se la programmazione è andata a buon fine.
- Se la programmazione è avvenuta si incrementa l'indice i, il quale individua l'indirizzo di memoria della word successiva (Address+i) e si ricomincia il ciclo fino al completamento della riga (i=128).

- Se la programmazione non è avvenuta, cioè c'è ancora qualche cella ad uno all'interno della word, si verifica se la tensione di programmazione impostata è già la massima consentita dalla memoria.
- Se non siamo ancora al valore massimo di V_{PR} questa si incrementa di uno step, fra quelli permessi dalla memoria e si effettua una nuova operazione di programmazione della sola word interessata dal fallimento a tensione maggiore. In questo caso, l'indice i non viene incrementato così da ricontrollare la medesima word.
- Se siamo al valore massimo di V_{PR} consentito (V_{PR_MAX}= 11,27V) si incrementa un contatore (k) e si effettua una nuova ed addizionale operazione di programmazione della sola word interessata dal fallimento. In questo caso, l'indice i non viene incrementato così da ricontrollare la medesima word.
- Dopo cinque operazioni consecutive di program con la V_{PR_MAX} (k=5) se vengono rilevati ancora errori nella programmazione, la sub-routine forza l'uscita e segnala il fail della cella.

La scelta di riprogrammare la sola word interessata dal fallimento e non tutta la riga è una scelta conservativa per applicare il minore stress possibile all'array.

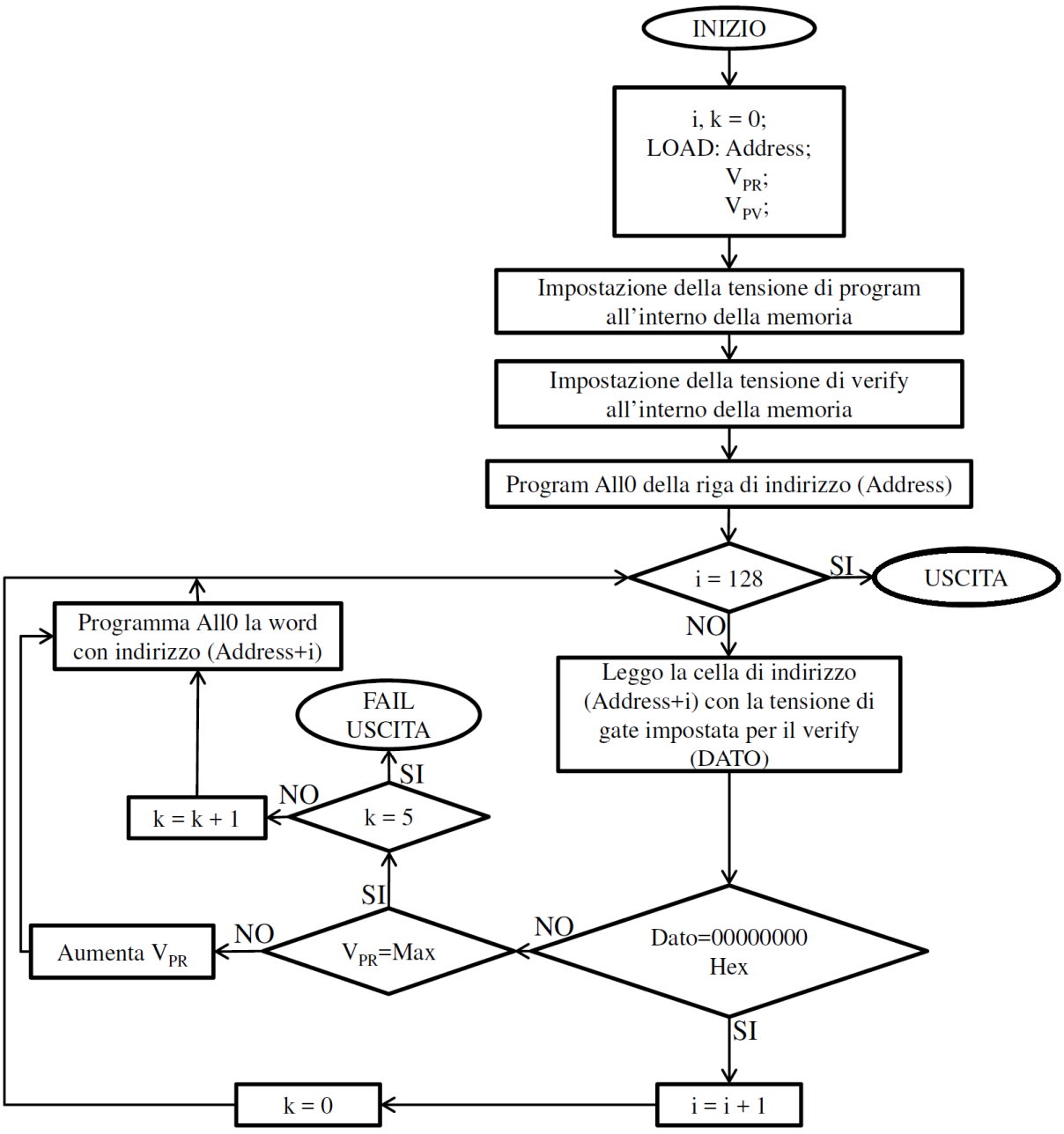


Fig. 42: Diagramma di flusso dell'algoritmo Program All 0.

4.4.2 Erase with verify

Nella figura 43 è riportato il diagramma di flusso della sub-routine “erase 0 with verify”. Essa riceve dalla routine principale le variabili seguenti:

- Address, identifica l’indirizzo di riga su cui deve essere effettuata l’operazione.
- V_{ER}, che imposta la tensione di bulk (6,52V÷11,27V) in cancellazione.
- $V_{EV,}$ imposta la tensione di control gate nella lettura per l’erase con verify. Tale tensione è minore del valore nominale di lettura della quantità pari al margine, di solito fra 1÷2V, che si vuole avere rispetto alla tensione rilevata nella coda delle distribuzioni delle celle cancellate.

Dopo aver caricato le variabili dalla routine principale, l’algoritmo cancellerà l’intera riga selezionata dall’indirizzo memorizzato tramite la variabile interna Address. Successivamente per 128 cicli, pari al numero di word che compongono la riga, si effettuerà l’operazione di program verify.

- Si legge la prima word della riga appena cancellata applicando la tensione di erase verify (V_{EV}) per effettuare la lettura. La tensione V_{EV}, impostata dall’utente nella fase iniziale della routine di ciclatura, è normalmente minore della tensione di read verify in modo da marginare la distribuzione delle celle cancellate rispetto alle condizioni standard di lettura.
- Si verifica se il contenuto delle celle della word letta è uguale a uno (FFFFFFFF Hex), cioè si controlla se la cancellazione è andata a buon fine.
- Se la cancellazione è avvenuta senza errori, si leggono i bit di ECC (Error Correction Code) e si verifica che anch’essi siano stati cancellati (all 1).
- Se la anche la cancellazione dei bit di ECC è avvenuta si incrementa l’indice i, il quale individua l’indirizzo di memoria della word successiva (Address+i) e si ricomincia il ciclo fino al completamento della riga (i=128).
- Se la cancellazione dei bit della word o dei bit di ECC non è avvenuta correttamente, cioè c’è ancora qualche zero all’interno, si verifica se la tensione di cancellazione impostata è già la massima consentita dalla memoria.
- Se non siamo ancora al valore massimo di V_{ER} questa si incrementa di uno step, fra quelli permessi dalla memoria stessa, e si esce dal ciclo per effettuare una nuova operazione di cancellazione dell’intera riga a tensione maggiore.

- Se siamo al valore massimo di V_{ER} consentito ($V_{ER\ \_MAX}$=11,27V) si incrementa un contatore (k), si esce dal ciclo e si effettua una nuova operazione di cancellazione dell'intera riga.
- Dopo dieci operazioni consecutive di erase con la $V_{ER\ \_MAX}$ (k=10) se la riga risulta ancora non cancellata correttamente la sub-routine forza l'uscita e segnala il fallimento.

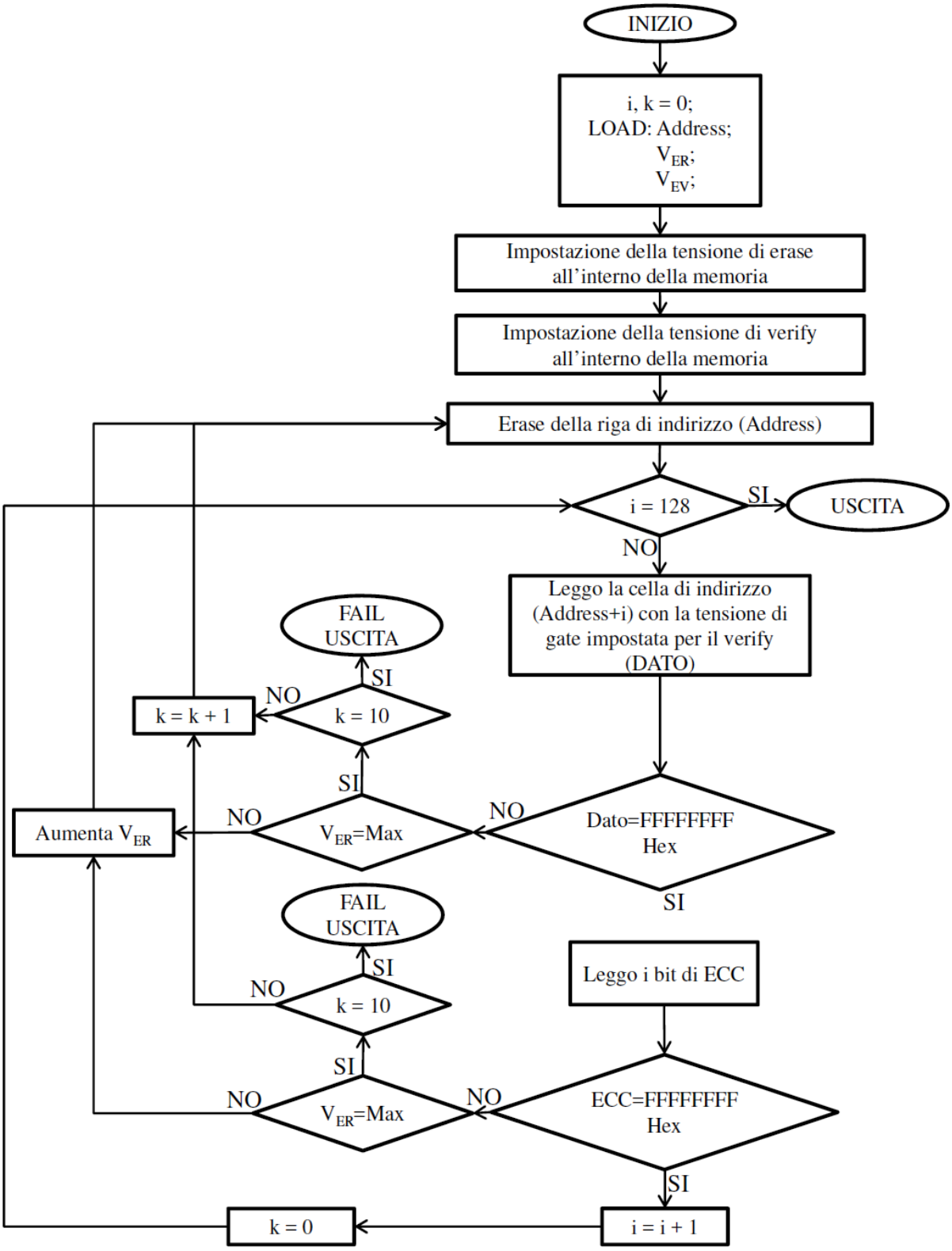


Fig. 43: Diagramma di flusso dell'algoritmo di "Erase con Verify".

4.4.3 Refresh

L'algoritmo di refresh, contenuto nella ciclatura completa, permette di rinfrescare, nel semi-array selezionato, un numero di righe N pari al numero di cicli previsto dalla ciclatura. L'operazione di refresh viene effettuata per riga, questa consiste nella lettura della stessa cella di memoria con due valori di tensione di control gate, quello standard, V_R, e quello di verify, V_{RV} con $V_R<V_{RV}$. Se esistono differenze fra la prima lettura e la seconda la cella verrà riscritta con i valori letti utilizzando la soglia più bassa.

Il software sviluppato gestisce in modo "intelligente" l'operazione di refresh, infatti mantiene all'interno dell'array di memoria l'indirizzo della prossima pagina da rinfrescare così che anche in condizioni critiche, quali interruzione di alimentazione o operazioni di refresh non concatenate, si garantisce il corretto funzionamento delle operazioni e si evita di rinfrescare accidentalmente sempre la stessa riga della memoria o che alcune righe non siano soggette a refresh per lungo tempo.

Nella figura 44 è riportato il diagramma di flusso della sub-routine di "refresh". Essa riceve dalla routine principale le seguenti variabili:

- Bench, identifica se stiamo operando nel semi-array superiore o inferiore (0=superiore; 1=inferiore).
- $V_{RV,}$ imposta la tensione di control gate nella lettura per il read verify. Tale tensione è maggiore del valore nominale di lettura della quantità pari al margine, di solito fra 0÷1V, che si vuole avere rispetto alla lettura standard, ma inferiore al valore $V_{PV.}$

Dopo aver caricato le variabili dalla routine principale l'algoritmo opera nel seguente modo:

- Attraverso un'operazione di lettura carica in memoria il Row_Counter (RC) che contiene l'indirizzo della riga da rinfrescare. Il Row_Counter è contenuto nelle prime sedici word dell'ultima riga del semi-array selezionato, quindi se si è nel semi-array superiore avrà indirizzo 0x0FF80h mentre se si è nel semi-array inferiore avrà indirizzo 0x1FF80h. Il Row_Counter contiene l'indirizzo della riga da rinfrescare in maniera codificata, infatti dato che un semi-array è composto da 512 righe e che sedici word (1 word=32 bit) contengono 512 bit si è deciso di azzerare un bit ad ogni refresh, partendo dalla prima riga. Per ricavare quindi il

numero della riga bisogna contare il numero di zeri contenuti nel Row_Counter. Moltiplicando quest'ultimo per il numero di word di una riga (128) e convertendolo in esadecimale, si ottiene l'indirizzo della riga da rinfrescare.

- Ricavato l'indirizzo della riga da rinfrescare si effettua la lettura dell'intera riga con i valori standard di tensione e si memorizzano tali valori all'interno di un array, di lunghezza peri a 128 word, denominato Row_0.

Successivamente per un numero di cicli pari a 128:

- Si effettua la lettura della word identificata dall'indirizzo RC+i con la tensione di verify impostata dall'utente (V_{RV}) e si memorizza nell'array Row_1[i].
- Si comparano, dunque, i due valori Row_0[i] e Row_1[i] e se sono uguali si procede all'esecuzione dell'algoritmo sulla word successiva incrementando l'indice i.
- Se i valori non sono identici, si riprogramma la word dell'indirizzo in uso (RC+i) con i valori letti in modalità user mode, quindi si effettua nuovamente la lettura con la tensione di verify, $V_{RV,}$ e si comparano nuovamente i valori.
- Al completamento delle word, i=128, si esce dal ciclo e si aggiorna il Row_Counter aggiungendo uno zero.
- Se il Row_Counter è pieno, cioè abbiamo rinfrescato la 551^{a} riga (indirizzo 0x0FF00 se in array superiore, indirizzo 0x1FF00 se in array inferiore), viene cancellato effettuando un erase di tutta la riga.

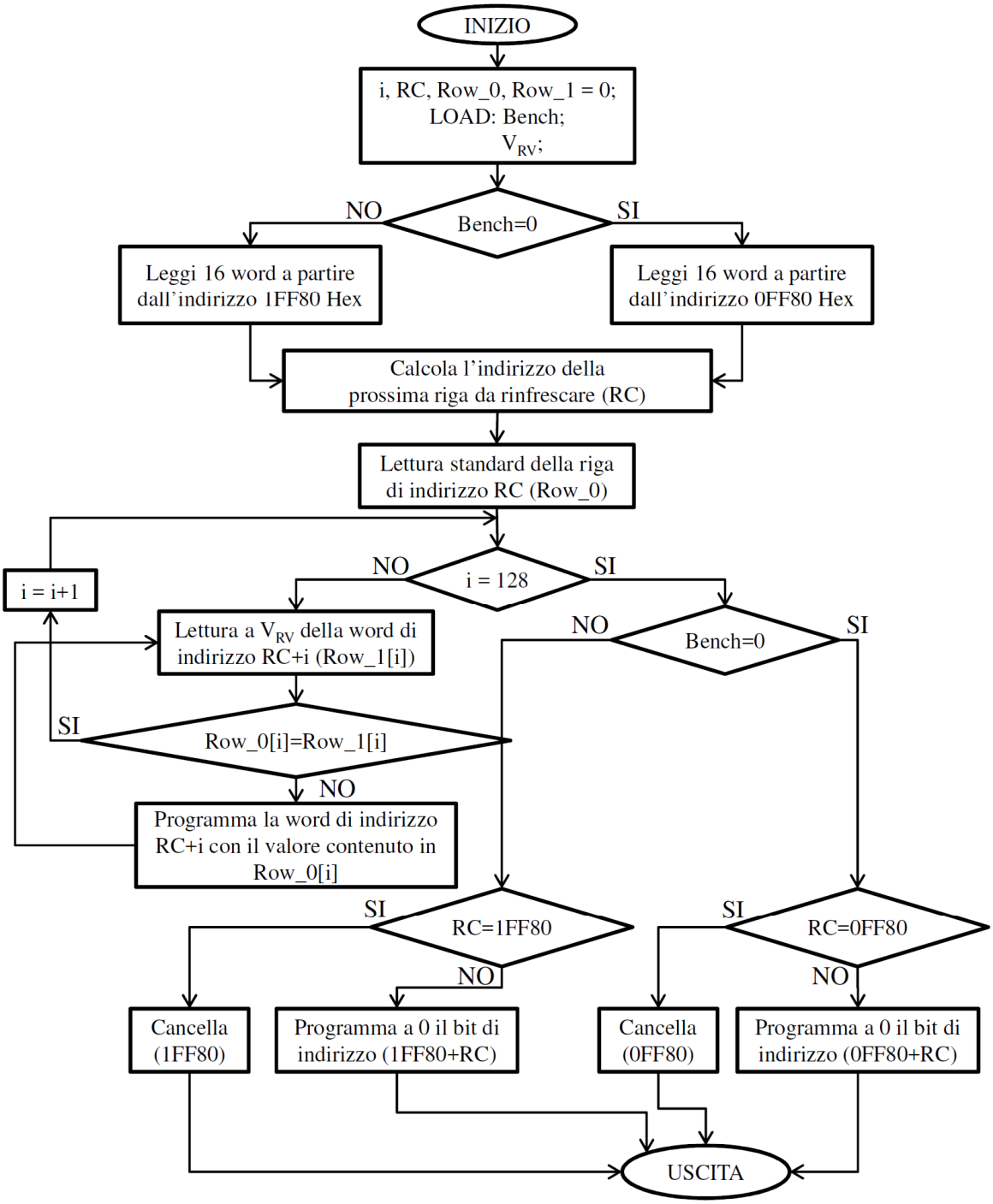


Fig. 44: Diagramma di flusso dell'algoritmo di refresh.

4.5 Analisi e Ottimizzazione degli Algoritmi di Gestione

Durante il testing delle memorie sono state effettuate diverse prove per valutare il tempo di vita della memoria, in particolare si è cercato di ottimizzare l'algoritmo di erase con verify. La scelta di ottimizzare l'algoritmo di erase è stata dettata da ben noti motivi dalla letteratura infatti, come riportato in Figura 45, la chiusura della finestra di erase è molto più pronunciata di quella di program.

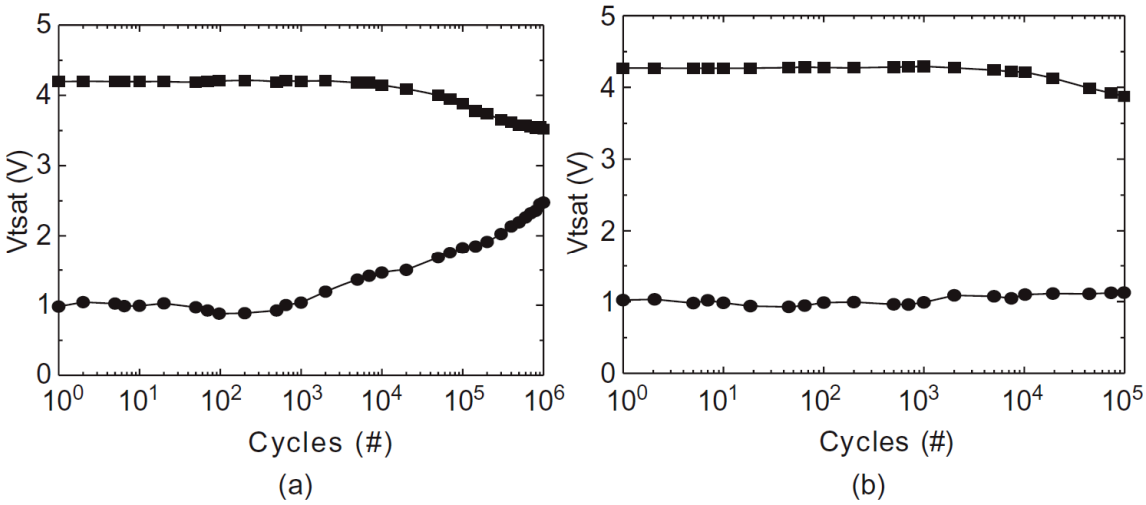


Fig. 45: Caratteristiche delle distribuzioni di soglia durante ciclatura.
A) tensione e durata fissate B) tensione fissa e numero di impulsi di erase/program variabile.
(Figura 1 e 2 di [12])

In particolare in figura 45 a) sono riportate le tensioni di soglie di una memoria Flash NOR all'aumentare dei cicli di programmazione/cancellazione effettuati a tensione e durata fissa senza alcun verify [12]. La chiusura della tensione di soglia di programmazione e della soglia di cancellazione è causata da diversi meccanismi. La chiusura della tensione di soglia di programmazione è dovuta alla cattura di elettroni nell'ossido vicino al drain, che diminuisce l'efficienza dell'iniezione di elettroni caldi (CHEI). La chiusura della soglia di cancellazione invece è dovuta a due fenomeni, la cattura (trapping) di elettroni nella regione di sovrapposizione fra source e gate causata dal FN tunnelling e la degradazione della transconduttanza dovuta ai danni della programmazione per elettroni caldi. Ripetuti cicli di FN tunnelling provocano l'intrappolamento di elettroni nella regione di sovrapposizione fra gate e source dove avviene il tunnelling e la presenza di questi elettroni riduce la corrente di tunnel e causa quindi la chiusura della finestra di cancellazione. La degradazione della transconduttanza invece riduce la corrente di lettura del canale e provoca la chiusura della finestra di tensione di soglia cancellata.

La figura 45 b) invece mostra una ciclatura effettuata con le stesse caratteristiche della 45 a) ma con numero di impulsi di programmazione e cancellazione variabili, quindi la durata della programmazione/cancellazione sarà adattata alle celle e al loro degradamento e, come si evince dalla minore chiusura delle caratteristiche, questo provoca una minore degradazione della cella stessa.

Partendo da queste premesse sono state effettuate diverse prove di ciclatura variando l'algoritmo di erase e cercando di trovare l'ottimo per il tempo di vita della memoria stessa.

Lo studio sperimentale è stato effettuato eseguendo 5 ciclature con 5 algoritmi differenti di erase utilizzanti la medesima finestra di tempo di erase, pari a 7,5ms, ed una programmazione a tensione fissa, pari a 10,75V. Si è iniziato con l'implementazione di un algoritmo non adattativo, denominato "Ciclatura 1", il cui diagramma di flusso è riportato in figura 46.

Ognuno dei diversi algoritmi è stato testato su 10 righe di memoria ed ogni riga è composta da 128 parole, ognuna con 32 bit di dati e 6 bit ECC (Error Correction Code), per cui si ottengono complessivamente 48640 bit per ogni sezione considerata (10 righe) e tutti sono stati eseguiti a partire da una memoria programmata in modo nativo con il codice ECC (Error Correction Code) abilitato, per cui abbiamo 46080 bit di dati programmati (tutti zero) e 7680 cancellati bit ECC (tutti).

L'algoritmo non adattativo (Ciclatura 1) è stato utilizzato come base i cui risultati possono essere raffrontati con gli altri algoritmi che verranno successivamente esposti. Esso inizia con l'impostazione dei seguenti parametri:

- Numero di cicli = N;
- La finestra di cancellazione massima pari a 2,5ms.
- Tensione di programmazione = 10,75V;
- Tensione di cancellazione massima = 11,27V;
- Numero di righe = 10;
- Riga di inizio = 0;
- Due variabili di conteggio interno = J e K.

Per il numero totale di cicli pari a N, viene eseguita una fase di programmazione di tutte le righe considerate, scrivendo tutti i bit a zero. Vengono quindi eseguiti tre cicli di cancellazione di tutte le righe considerate per raggiungere la finestra di cancellazione di 7,5ms.

Quindi, viene eseguita una verifica di cancellazione e se la fase di verifica è correttamente superata, il ciclo continua, altrimenti viene constatato il fallimento della cancellazione e viene stampato il numero del ciclo.

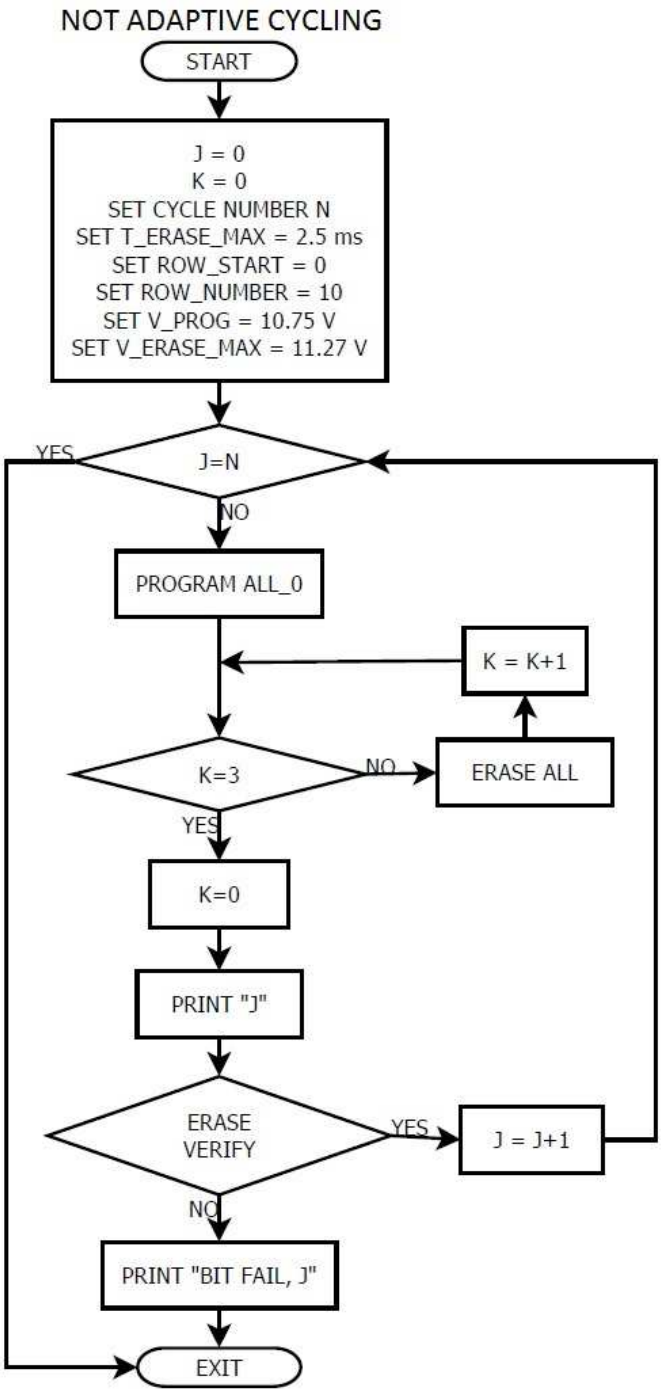


Fig. 46:Diagramma di flusso dell'algoritmo di "Ciclatura 1" non adattativo.

I risultati di questo test hanno mostrato, come riportato in figura 47, una costante degradazione della tensione di soglia all'aumentare del numero dei cicli, in particolare si è avuto un fallimento nella cancellazione dopo 759016 cicli.

Questo valore verrà usato come termine di paragone per la valutazione della bontà dei successivi algoritmi adattativi.

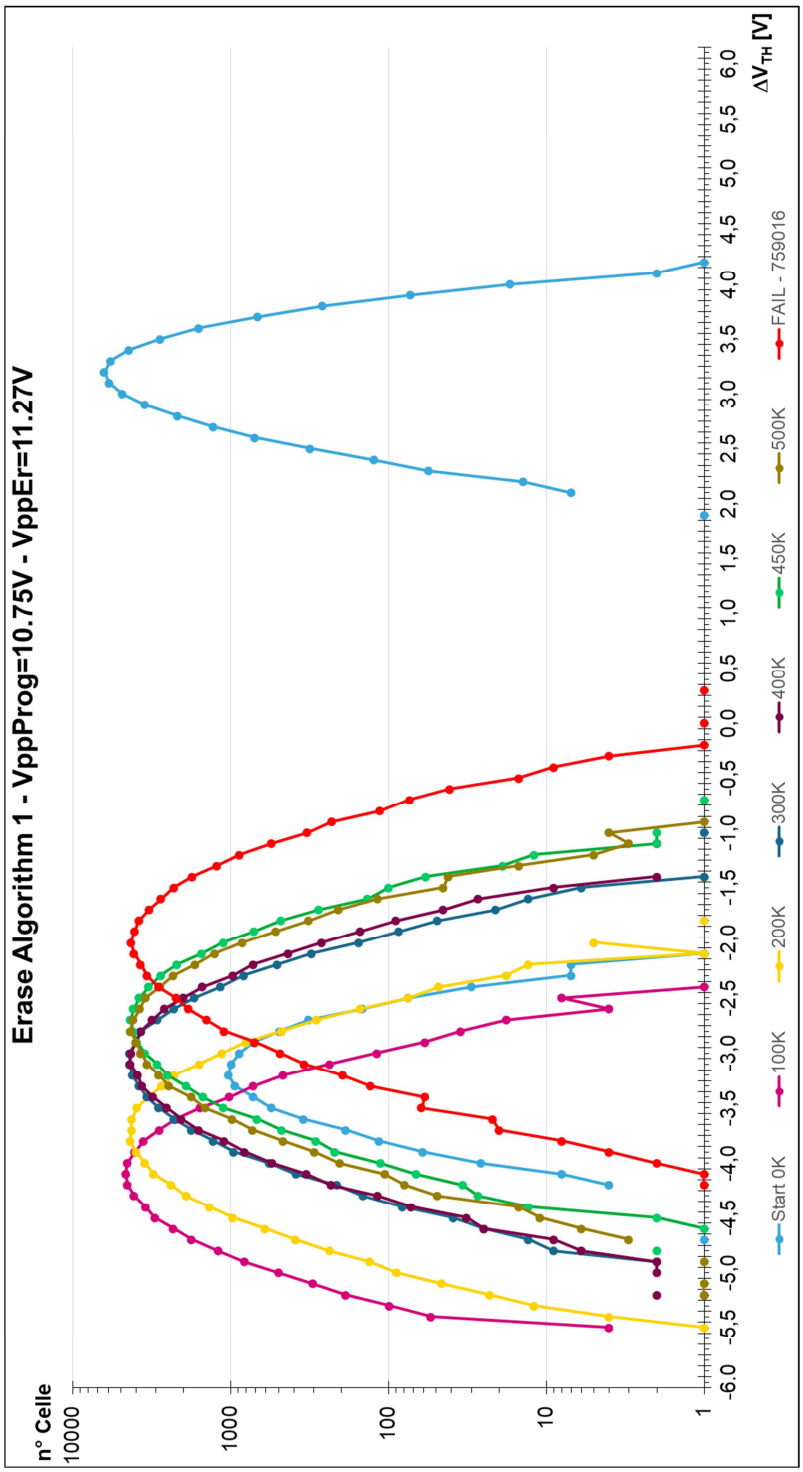


Fig. 47: Risultati dell'algoritmo ci CIclatura non adattativo, con tensione di program di 10,75V e tensione di erase di 11,27V.

Il ciclo, il cui diagramma di flusso è mostrato in figura 48, ha caratteristiche di adattabilità e quindi determina un accumulo di stress ed una conseguente lifetime della memoria dipendente dalle caratteristiche di tale adattamento. Esso inizia con l'impostazione dei seguenti parametri:

- Numero di cicli = N;
- Finestra di cancellazione del tempo = 750µs;
- Tensione di programmazione = 10,75V;
- Tensione di cancellazione iniziale, V_{ER};
- Numero di righe da ciclare, Row_Number = 10;
- Riga di inizio = 0;
- Due variabili di conteggio interno, k e i = 0.
- Una variabile di verifica del fallimento, Fail = 0.

Per un numero totale di cicli pari a N, viene eseguita una fase di programmazione di ogni riga scrivendo tutti i bit a zero, quindi viene cancellata una riga alla volta, attraverso una opportuna routine.

In base alla tipologia dell'algoritmo di ciclatura implementato, la routine di cancellazione con verify, che segue l'architettura mostrata nella figura 43 del paragrafo 4.4.2, applicherà tensioni di valore differente ed un numero di impulsi variabile in funzione del successo/fallimento della relativa operazione di cancellazione, avendo un massimo di ripetizioni dell'impulso pari a 10. Se al decimo impulso dovesse avvenire un ulteriore fail, la routine esce e memorizza la riga in cui è avvenuto il fallimento e il numero di cicli effettuati.

Il numero di impulsi è pari a 10 per garantire una finestra di cancellazione totale pari a 7,5ms identica a quella dell'algoritmo non adattativo "Ciclatura 1".

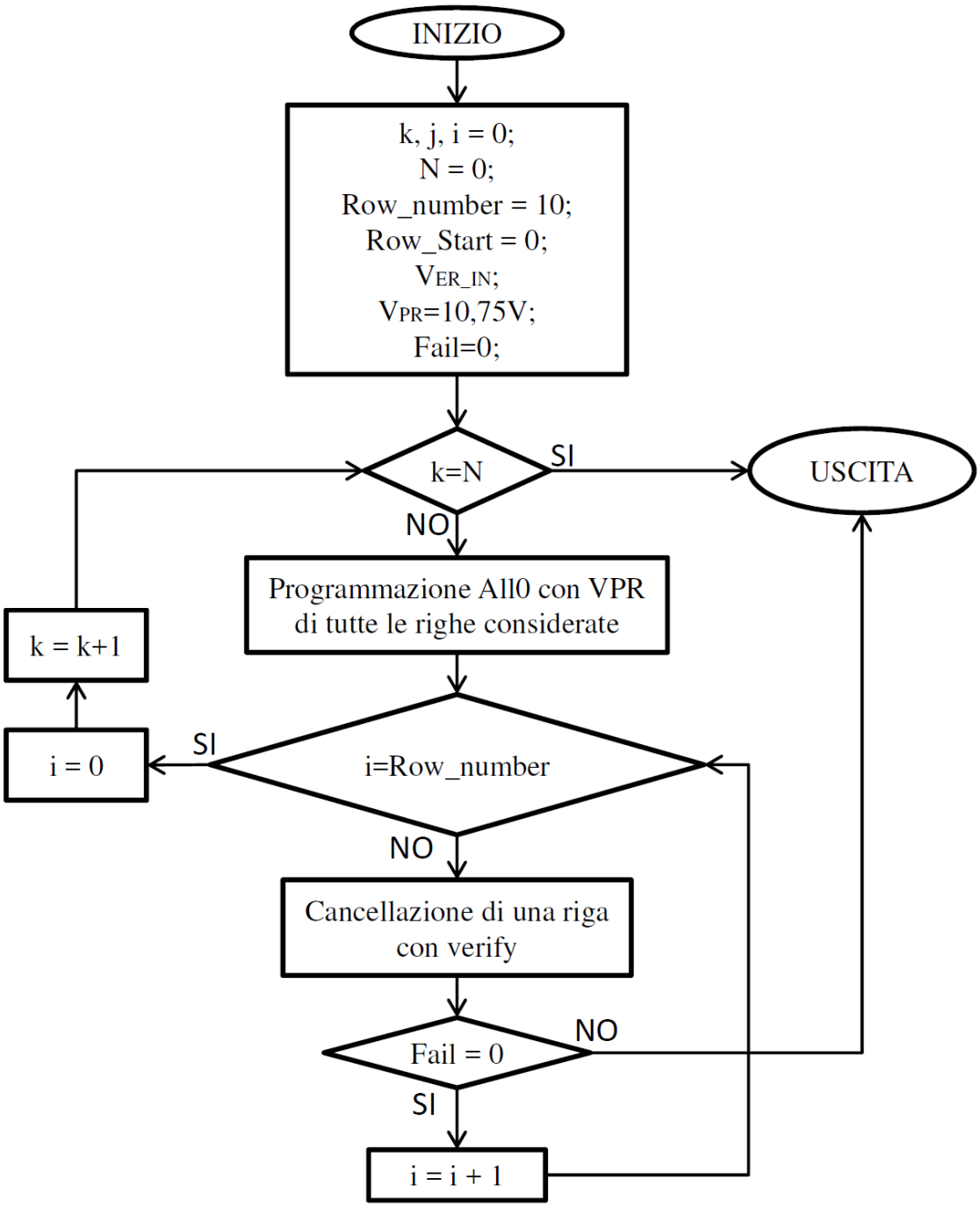


Fig. 48: Diagramma di flusso delle ciclature adattarive.

Sono stati in effetti implementati e sperimentati quattro differenti algoritmi di tipo adattativo differenziandosi per la tipologia di tensioni di cancellazione applicate (vedi figura 49).

La “ciclatura 2” presenta una tensione di erase iniziale di 8,64V, che aumenta linearmente all’aumentare del numero di impulsi e con un plateau dal quinto al decimo impulso pari alla tensione massima di 11,27V.

La “ciclatura 3” presenta una tensione iniziale di 8,20V con un incremento più graduale ed un plateau pari alla tensione massima di 11,27V dall’impulso otto al dieci.

La “ciclatura 4” ha una tensione iniziale pari al minimo consentito di 6,52V e un incremento esponenziale con un plateau pari alla tensione massima di 11,27V dall’impulso otto al dieci.

La “ciclatura 5” è stata elaborata a partire dalle ciclature già effettuate e che hanno fornito i risultati migliori. In particolare si vedrà che le ciclature più performanti sono la 2 e la 4 e quindi si è scelto un valore iniziale più alto e pari a 9,32V ed un incremento più graduale fino al plateau pari alla tensione massima di 11,27V dall’impulso otto al dieci.

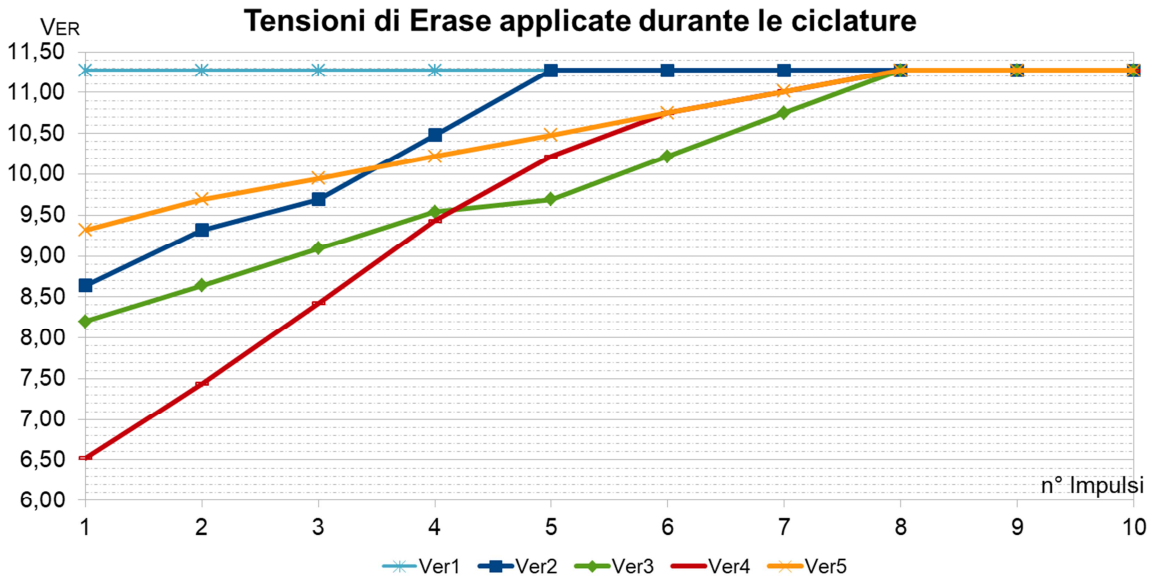


Fig. 49: Tipologie di tensioni applicate ai differenti algoritmi di ciclatura.

I risultati delle ciclature sono mostrati nelle figure 50, 51, 52, 53. Essi evidenziano un notevole miglioramento in termini prestazionali del tempo di vita della memoria in termini di cicli di programmazione/cancellazione, con incrementi della vita della memoria di circa il 260% (ciclatura 2), 155% (ciclatura 3), 185% (ciclatura 4), 222% (ciclatura 5); questo a causa di un minor stress e degrado subito dalla memoria grazie all’utilizzo di algoritmi più elaborati.

Tutte le ciclature adattative evidenziano una distribuzione iniziale della tensione di soglia inferiore a quella non adattativa, poiché è stato applicato un algoritmo intelligente che ha un livello iniziale di tensione di cancellazione minore.

Si può notare che le distribuzioni rimangono mediamente allo stesso valore centrale poiché l'azione d'invecchiamento, che tipicamente spinge verso destra le distribuzioni delle celle cancellate, è stata compensata dall'aumento di tensione di cancellazione imposto dinamicamente dall'algoritmo quando necessario.

La ciclatura 2 che ha un valore di tensione di cancellazione iniziale pari a 8,64V, un incremento lineare ed un plateau di 5 impulsi ha dato dei risultati, figura 51, veramente degni di nota, effettuando ben 1988059 cicli di programmazione/cancellazione (P/E), cioè circa il 260% in più dell'algoritmo non adattativo.

Dopo i risultati della ciclatura 2 nella ciclatura 3 si è ipotizzato di diminuire ulteriormente la fonte di stress, impostando il valore di tensione di erase iniziale al valore 8,20V e diminuendo l'incremento della stessa fra un impulso ed un altro riducendo il plateau da 5 a 3 impulsi. I risultati, figura 52, mostrano invece un decadimento delle prestazioni rispetta alla precedente, 1178433 cicli di P/E. si è ipotizzato che questo fosse dovuto all'aumento del numero totale di impulsi subito dalla memoria causato dal veloce incremento degli impulsi subito all'inizio a causa delle basse tensioni.

Nella ciclatura 4 si è provato un incremento delle tensioni di cancellazione di tipo esponenziale così da avere degli incrementi minori negli ultimi impulsi e maggiori all'inizio, così da cercare di compensare il degradamento iniziale verificatosi nella ciclatura 3, il veloce incremento del numero di impulsi dovuto al basso incremento di tensione. Questa ciclatura ha dato dei discreti risultati portando il tempo di vita a 1410643.

Basandoci sui risultati ottenuti con le ciclature precedenti, si è sviluppata la ciclatura 5 che ha un valore iniziale di tensione maggiore dei precedenti e degli step più piccoli ed uniformi, questa ha ottenuto un tempo di vita pari a 1690594, il 222% in più della non adattativa.

Da questi test risulta evidente che lo stress non dipende solamente dal valore di tensione assoluto imposto ai capi della cella, ma anche da come gli incrementi della tensione vengono gestiti e quindi dal numero totale di impulsi subiti dalla memoria, un integrale fra valore assoluto imposto, tempo di applicazione e numero totale di impulsi subiti.

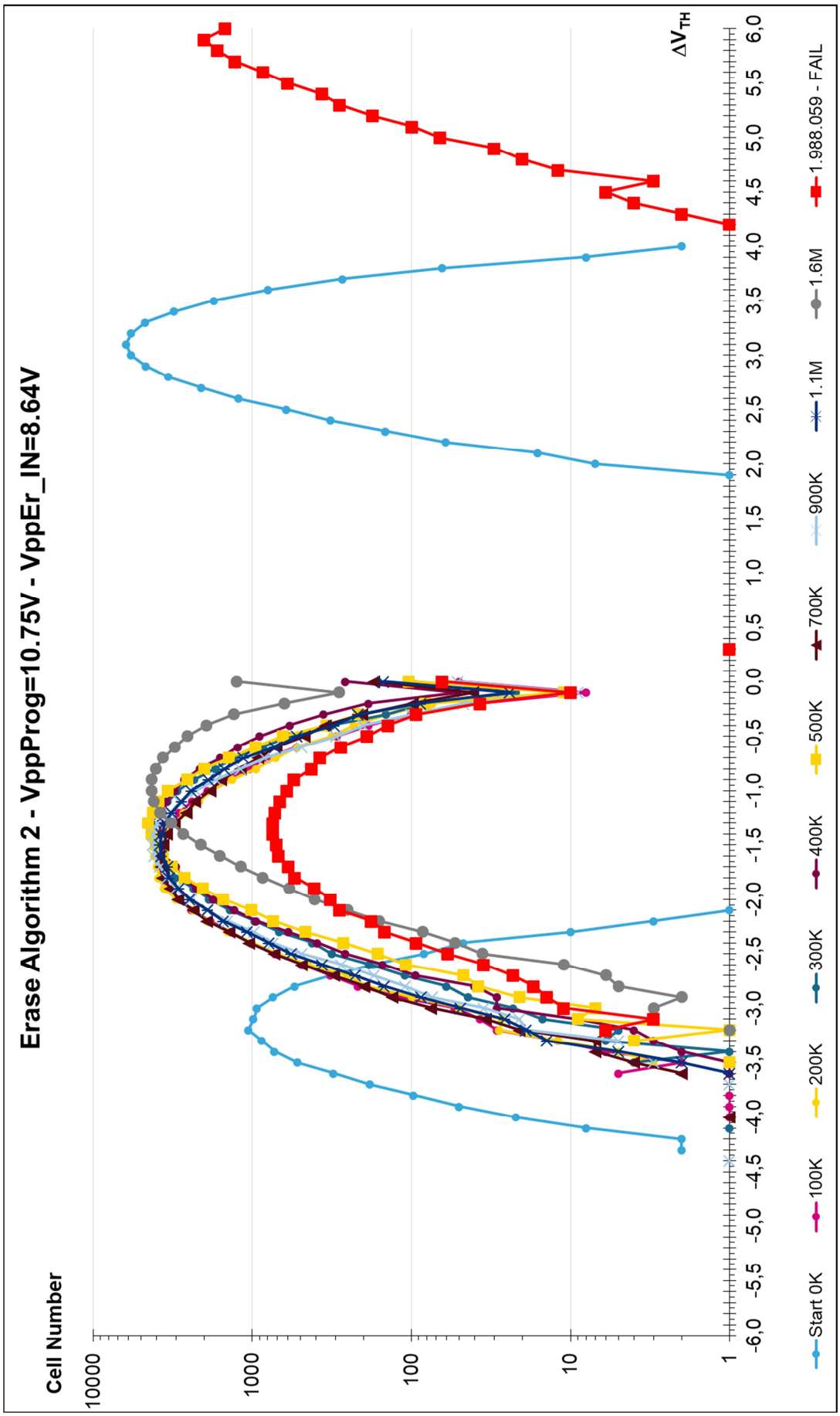


Fig. 50: Risultati della ciclatura 2.

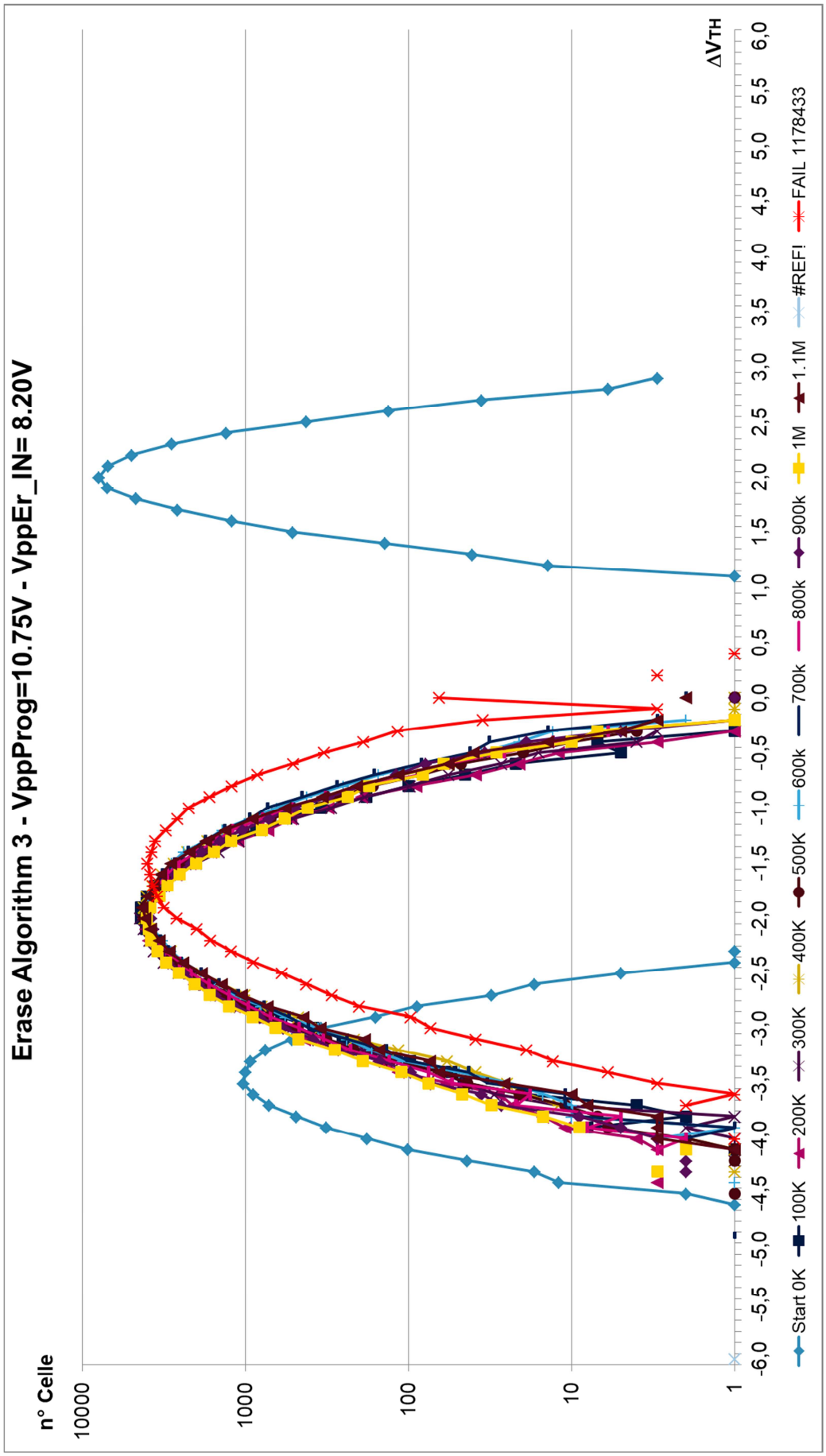


Fig. 51: Risultati della ciclatura 3.

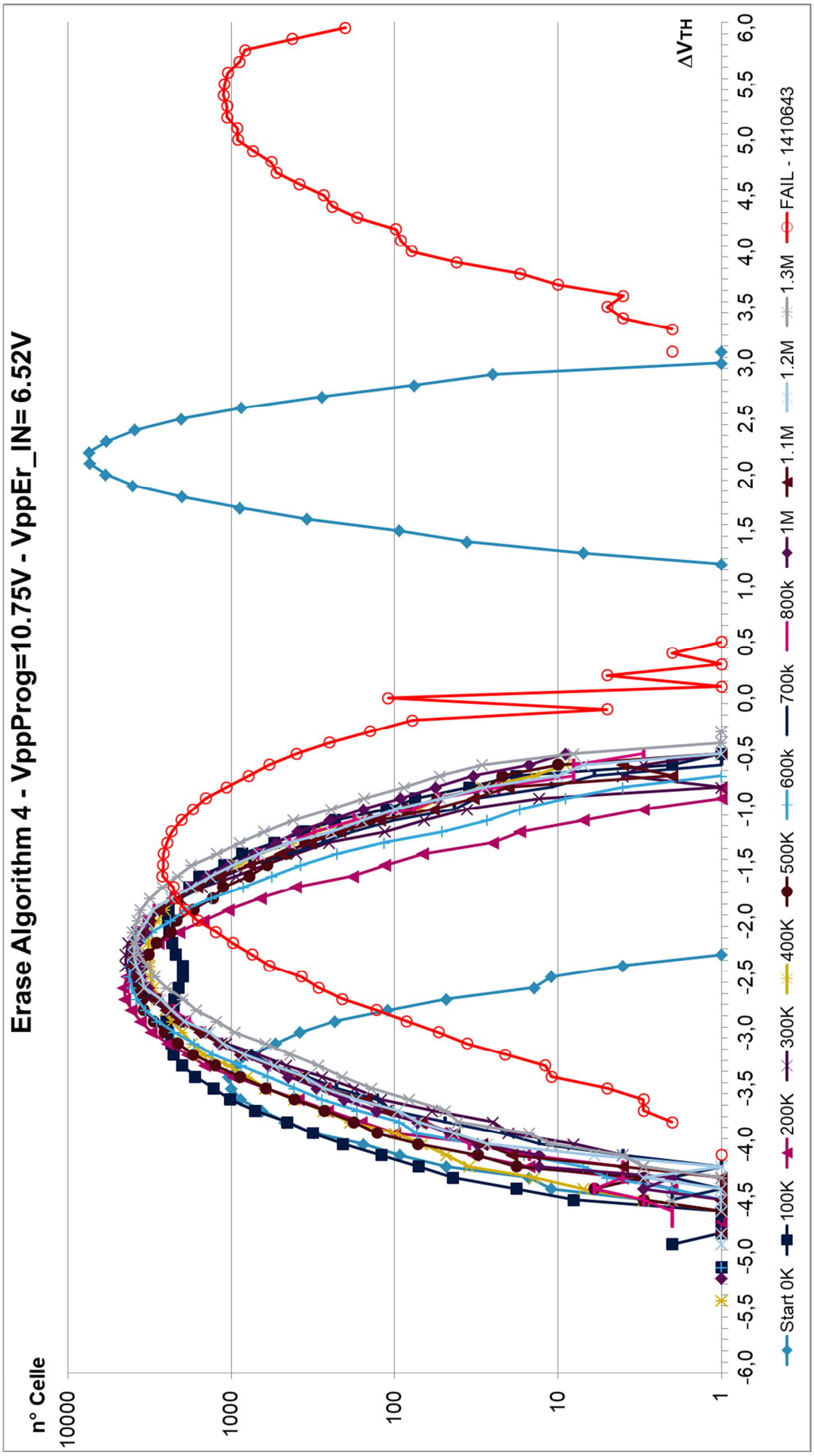


Fig. 52: Risultati della ciclatura 4.

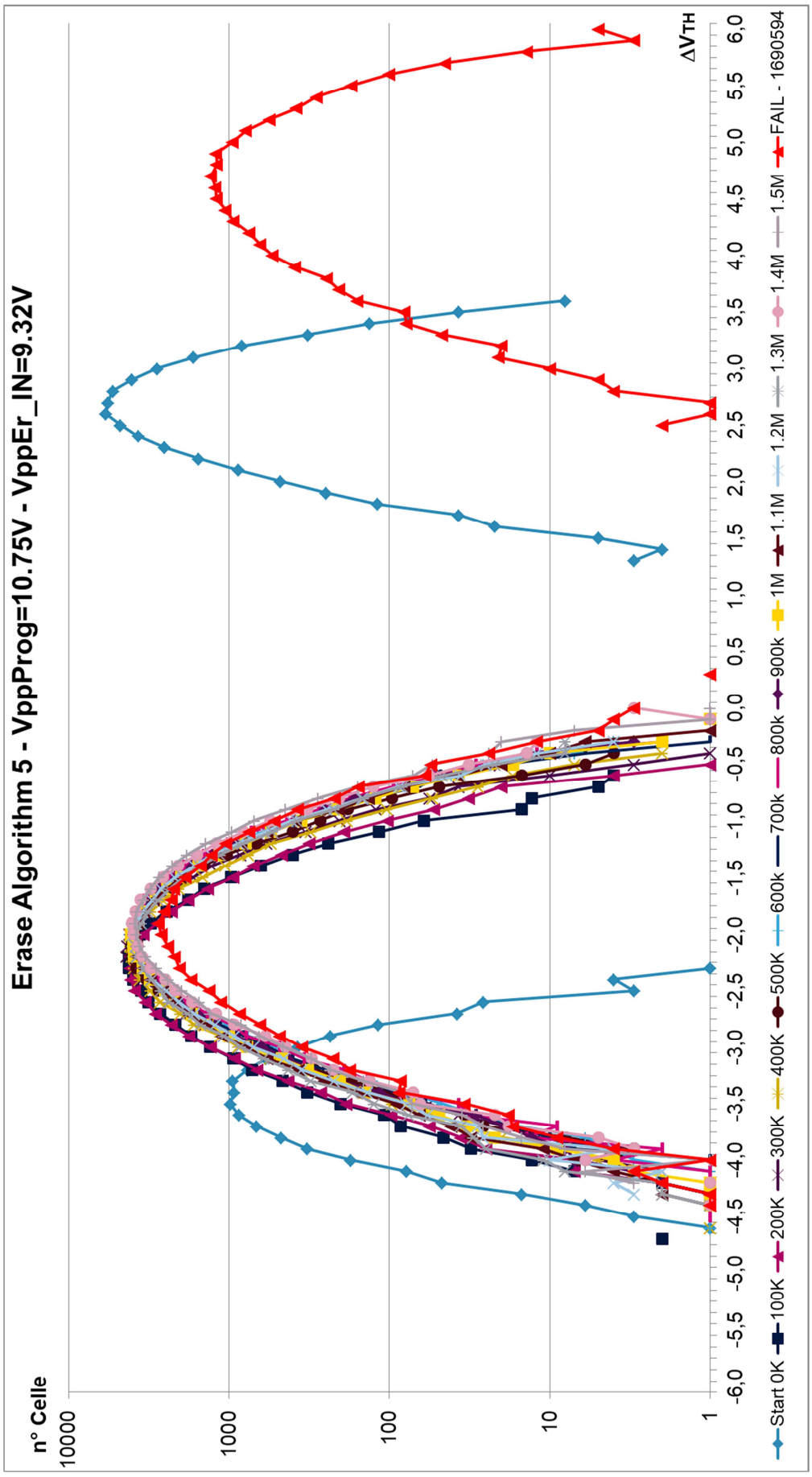


Fig. 53: Risultati della ciclatura 5.

5 Conclusioni

Sono state analizzate le principali cause di stress degli array e mostrate le principali tecniche di test normalmente utilizzate, Automated Test Equipment (ATE), Built In Self Test (BIST) e le innovative tecniche Software Based Self Test.

Si è sviluppata una scheda a basso costo, denominata Portable-ATE, per il testing di un singolo memory test chip alla volta, il cui hardware a basso costo e il software user friendly ne fanno un validissimo strumento di sviluppo, indispensabile al progettista di memorie.

Con tale dispositivo è stata possibile la caratterizzazione e lo studio di una nuova tipologia di memoria di STMicroelectronics. L'innovazione del Portable ATE è la completa e quasi esaustiva possibilità di testing della memoria senza l'utilizzo di ATE classici, in particolare sono stati effettuati test di drain stress, gate stress e test di ciclatura. Inoltre, è stato possibile lo sviluppo ed il test diretto in memoria degli algoritmi di gestione quali program con verify, erase con verify e refresh pur non essendo presente all'interno del memory test chip il microprocessore o il Program-Erase-Controller (PEC) completo. È stata inoltre effettuata l'analisi dell'impatto di diverse tipologie di algoritmi per l'erase con verify sul tempo di vita della memoria con risultati entusiasmanti.

La grandezza del piccolo Portable ATE sta nella flessibilità e nella potenza di calcolo, con cui si possono davvero effettuare quasi tutti i test di un ciclo EWS.

A tal proposito, ulteriori miglioramenti potrebbero essere apportati al Portable-ATE per renderlo più completo ed adattabile. In particolare, si potrebbe aggiungere una cella peltier comandata dal microprocessore per effettuare prove al variare della temperatura. Sarebbe auspicabile l'utilizzo di opportuni multiplexer o di un layout differente in modo da rendere flessibili i collegamenti dal microprocessore al package della memoria, così da rendere la scheda utilizzabile per test chip con piedinature differenti (a patto che necessitino degli stessi segnali di controllo).

Riguardo la valutazione degli stress, ulteriori indagini sono possibili per riuscire a realizzare algoritmi intelligenti e adattabili senza bisogno di hardware aggiuntivo, ma sfruttando il microprocessore, questa sarebbe un'ulteriore evoluzione dei così detti SBST.

6 Bibliografia

1. F. Masuoka, M. Asano, H. Iwahashi, T. Komuro and S. Tanaka, *"A New Flash E2PROM Cell Using Triple Polysilicon Technology",* 1984 International Electron Devices Meeting, pp. 464-467, 1984.

2. D. Kahng and S. M. Sze, *"A Floating Gate and Its Application to Memory Devices"* Bell Syst. Tech. J., 1967.

3. F. Masuoka, M. Momodomi, Y. Iwata and R. Shirota, *"New ultrahigh density EPROM and flash EEPROM with NAND structure cell"* 1987 International Electron Devices Meeting, pp. 552-555, 1987.

4. Joe E. Brewer, Manzur Gill, *"Nonvolatile Memory Technologies with Emphasis on Flash: A Comprehensive Guide to Understanding and Using NVM Devices"*, Wiley-IEEE Press, 2008. *"NAND vs. NOR Flash Memory Technology Overview"*, Toshiba, 2006.

5. B. Eitan and D. Frohman-Bentchkowsky, *"Hot Electron Injection into the Oxide in n-channel MOS-Devices"*, IEEE Trans. Electron Devices, Vol. ED-28, p.328, 1981.

6. M. Lenzlinger and E.H. Snow, *"Fowler–Nordheim Tunneling into Thermally Grown SiO_2"*, J. Appl. Phys., Vol. 40, p.278, 1969.

7. B. Kaminska and K. Arabi, *"Mixed signal DFT: a concise overview"* ICCAD-2003. International Conference on Computer Aided Design pp. 672-679, San Jose, CA, USA, 2003.

8. E. I. Vatajelu; P. Prinetto; M. Taouil; S. Hamdioui, *"Challenges and Solutions in Emerging Memory Testing"* in IEEE Transactions on Emerging Topics in Computing , 2017.

9. P. H. Bardell, W. H. McAnney, and J. Savir, *"Built-In Test for VLSI: Pseudorandom Techniques"*, John Wiley and Sons, 1987.

10. M. Bushnell and V.D. Agrawal, *"Essentials of Electronic Testing for Digital, Memory & Mixed-Signal Circuits"*, Kluwer Academic Publishers, 2000.

11. P. Nicosia and F. Nava, *"Test Strategies on Non Volatile Memories Electrical Wafer Sort on NAND, NOR Flash and Phase Change Memories"*, 22nd IEEE Non-Volatile Semiconductor Memory Workshop, Monterey, CA, 2007.

12. P. Papavramidou and M. Nicolaidis, *"Transparent BIST for ECC-Based Memory Repair"*, On-Line Testing Symposium (IOLTS), 2013 IEEE 19th International, July 8-10, 2013

13. M. Nicolaidis, *"Theory of Trasparent BIST for RAMs"*, IEEE Transactions on Computer, Vol.45, No.10, October 1996.

14. M. Psarakis, D. Gizopoulos, E. Sanchez and M. Sonza Reorda, *"Microprocessor Software-Based Self-Testing"*, IEEE Design & Test of Computers, vol. 27, no. 3, pp. 4-19, May-June 2010.

15. A. van de Goor, S. Hamdioui and G. Gaydadjiev, *"Using a CISC microcontroller to test embedded memories"*, 13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, Vienna, Austria, 2010.

16. A. van de Goor, G. Gaydadjiev and S. Hamdioui, *"Memory testing with a RISC microcontroller"*, 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, 2010.

17. P. Bernardi, L. Ciganda, M. S. Reorda and S. Hamdioui, *"SW-based transparent in-field memory testing"*, 16th Latin-American Test Symposium (LATS), Puerto Vallarta, 2015.

18. W. J. Perez H., J. V. Medina, D. Ravotto, E. Sanchez and M. S. Reorda, "Software-Based Self-Test Strategy for Data Cache Memories Embedded in SoCs", 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Bratislava, 2008.

19. J. C. Yeh, K. L. Cheng, Y. F. Chou and C. W. Wu, *"Flash Memory Testing and Built-In Self-Diagnosis With March-Like Test Algorithms"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no. 6, pp. 1101-1113, June 2007.

20. J. Chen, N. Radjy, S. Cagnina, and J. Lien, *"Degradation Mechanism of Flash EPROM Program/Erase Endurance"*, 13th Annual IEEE Non-Volatile Semiconductor Memory Workshop, Monterey, CA, 1994.

21. S. Satoh, G. Hermink, K. Hatakeyama and S. Aritome, *"Stress-Induced Leakage Current of Tunnel Oxide Derived from Flash Memory Read Disturb Characteristics"*, IEEE Trans. Electron Devices, Vol. 45, No. 2, pp. 482–486, 1998.

7 Appendice – Attività Scientifica

7.1 Articoli su riviste scientifiche internazionali con processo di review (ISI):

1. G. Alieri, G. C. Giaconia, L. Mistretta, F. La Rosa, A. A. Cimino, *"Performance Evaluation of Non Volatile Memories with a Low Cost and Portable Automatic Test Equipment"* Accepted publication of the proceedings on the dedicated Springer LNEE book ApplePies 2016, September 15-15, 2016, Rome Italy.

7.2 Articoli presentati a congressi nazionali e/o internazionali:

1. G. Alieri, G. C. Giaconia, F. La Rosa, A. A. Cimino, *"Portable Automatic Test Equipment for Performance Evaluation of Non Volatile Memory"* Proceeding of GE2016 , June 22-24, 2016 – Brescia Italy.
2. G. Alieri, G. C. Giaconia, L. Mistretta, F. La Rosa, A. A. Cimino, *"Performance Evaluation of Non Volatile Memories with a Low Cost and Portable Automatic Test Equipment"* Proceeding of ApplePies 2016, September 15-15, 2016, Rome Italy.
3. G. Alieri, G. C. Giaconia, L. Mistretta, F. L. Rosa and A. A. Cimino, *"Impact of the erase algorithms on flash memory lifetime" 2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Giardini Naxos - Taormina, Italy, pp. 357-360, 2017.
4. G. Alieri, G. C. Giaconia, L. Mistretta, F. L. Rosa and A. A. Cimino, *"Impact of the erase algorithms on flash memory lifetime",* SIE 2017, Palermo, Italy, 21-23 Giugno 2017.

7.3 Brevetti internazionali:

1. Francesco LA ROSA, Gineuve ALIERI; STMicroelectronics (Rousset) SAS; *Procédé de gestion d'une ligne de bits défectueuse du plan mémoire d'une mémoire non volatile et dispositif de mémoire correspondant*; Patent - ST Ref: 15-RO-0335FR01, 2015;
2. Francesco LA ROSA, Gineuve ALIERI; STMicroelectronics (Rousset) SAS; *Procédé de gestion d'une ligne défectueuse du plan mémoire d'une mémoire non volatile et dispositif de mémoire correspondant*; Patent - ST Ref: 15-RO-0334FR01, 2015;
3. Francesco LA ROSA, Gineuve ALIERI; STMicroelectronics (Rousset) SAS; *NVM dynamic sense amplifier with «floating core»*; Patent - ST Ref: 16-RO-, 2016;