An Early-Life NAND Flash Endurance Prediction System

BARRY FITZGERALD¹, CONOR RYAN², AND JOE SULLIVAN.¹

¹Department of Electrical and Electronic Engineering, Limerick Institute of Technology, Moylish Park, Limerick, V94 EC5T, Ireland
²Department of Computer Systems, University of Limerick, Limerick, V94 T9PX, Ireland

Corresponding author: Barry Fitzgerald (e-mail: barry.fitzgerald@lit.ie).

ABSTRACT NAND flash memory – ubiquitous in today’s world of smart phones, SSDs (solid state drives), and cloud storage – has a number of well-known reliability problems. NAND data contains bit errors, which require the use of error correcting codes (ECCs). The raw bit error rate (RBER) increases with program-erase (P-E) cycling, and the number of P-E cycles the device can withstand before the RBER exceeds the ECC capability is called its endurance. ECC operates on data stored in a sector of NAND, and there is a large variation in the endurance of sectors within a device and across devices, resulting in excessively conservative endurance specifications. This research shows, for the first time, that a sector’s true endurance can be predicted with remarkable accuracy, using a combination of the sector’s location within the device, and measurements taken at the very beginning of life. Real-world data is gathered on millions of NAND sectors using a custom-built test platform. Optimised machine learning classification models are built from the raw data to predict if a sector will pass or fail to a fixed ECC threshold, after a target P-E cycling level has been reached. A novel technique is demonstrated that uses different ECC thresholds for model training and testing, which allows the models to be tuned so that they never misclassify samples that would fail. This eliminates ECC failures and data loss, allowing simpler, less expensive ECC schemes to be used for modern NAND devices. It also enables significant endurance extensions to be achieved.

INDEX TERMS NAND flash, solid state drives, endurance, retention, machine learning, gradient boosting, classification, prediction.

I. INTRODUCTION

NAND flash is a type of non-volatile memory that has seen an explosion in growth over the last 25 years, as the world’s data storage requirements have grown exponentially. It has been the fastest growing market in the history of semiconductors, with revenues of over US$52 billion in 2020 [1]. This growth has been driven by markets such as smart phones, media players, USB sticks and solid-state drives (SSDs), which have steadily been replacing hard disk drives (HDDs), both in personal computers and high-performance enterprise and data centre applications [12]. Flash-based storage offers many advantages over HDDs, such as faster performance and lower power consumption. However, NAND flash has a significant disadvantage: its reliability degrades with usage.

Data read back from a NAND device contains bit errors, requiring the use of error correcting codes (ECCs) by the NAND controller. The ECC engine operates on chunks of data called a codeword. For each codeword, the fraction of bits in error is known as the raw bit error rate (RBER), and if the RBER exceeds the capability of the ECC engine, an uncorrectable error occurs. As the NAND device is programmed and erased (known as P-E cycling), the NAND cells wear out and the RBER increases, until eventually the ECC engine is overwhelmed. Expressed in number of P-E cycles, this is known as the endurance limit. RBER also increases with storage time, resulting in a data retention limit. The data retention requirement of NAND devices is generally fixed, depending on application (one year for consumer/client applications, three months for enterprise applications [41]).

Maximum RBER occurs when the device has been P-E cycled to its endurance limit and the maximum allowable retention time has elapsed. NAND manufacturers set their endurance specifications to ensure that RBER will be below a target value at this end-of-life point. However, across a population of codewords, the RBER will vary considerably. To avoid uncorrectable errors, excessively conservative endurance specifications are chosen, based on worst-case
codeword RBER. Currently, NAND controllers operate on the assumption that codeword RBER is random and non-deterministic.

However, in this paper we show that this is not the case. Instead, codeword RBER can be predicted with a high degree of accuracy, based on a combination of two factors: a) wafer fabrication processing variations, which can be identified early in life, and b) NAND device architecture, and consequently the memory location in which a codeword is stored, known as a sector.

The purpose of this paper is to investigate if the RBER associated with each sector at the end of life (maximum endurance and retention) can be predicted, based on a combination of sector address and measurements taken at the beginning of life (after a few P-E cycles). Using a custom-built test platform, data is gathered from millions of sectors across 45 NAND devices. Machine learning models are then built from this data, to classify sectors as passing or failing at end of life, based on a fixed RBER threshold. Various model optimisation techniques are investigated, and a method to ensure the correct classification of failing sectors with 100% accuracy is demonstrated.

The remainder of this paper is structured as follows: Section II provides a comprehensive overview of NAND flash as it pertains to this work. Related prior work is discussed in Section III. Section IV outlines the motivation behind the research and the specific research questions that are answered by this work. An overview of the experimental design is presented in Section V, which consists of a data collection phase, a raw data analysis phase, and a machine learning phase. These three phases are discussed in Sections VI, VII, and VIII respectively. Section IX presents the results from the initial machine learning models, while Section X discusses a method to optimise model performance. Section XI introduces a novel model tuning technique, and the results from the final tuned prediction system are presented in Section XII, which outlines the endurance/capacity trade-off achievable if such a system were deployed. Finally, conclusions are presented in Section XIII.

II. NAND FLASH MEMORY
A. FLASH CELL

NAND devices are comprised of millions of flash cells. Flash cells are based on the metal-oxide-semiconductor (MOS) transistor, which consist of three terminals: source, drain and gate. If the voltage on the gate is above a threshold voltage \( V_T \), the transistor turns on and current can flow from source to drain. Flash cells have an additional floating gate embedded in the insulating oxide layer which separates the gate (called the control gate in a flash cell) from the transistor channel [23]. A cross-section of a flash cell is illustrated in Fig. 1.

Programming a flash cell involves tunneling electrons through the oxide layer onto the floating gate, where they are held by the insulating oxide [29]. This has the effect of raising the threshold voltage of the cell. Conversely, erasing a flash cell involves removing electrons from the floating gate, which lowers the threshold voltage. Reading a cell involves determining whether the \( V_T \) is high (programmed) or low (erased) by applying an intermediate voltage to the control gate and sensing if the cell is turned on or off (by detecting if current flows through the cell).

B. TYPES OF NAND

Original flash devices, known as single level cell (SLC) devices, stored one bit per cell [24]. For a population of cells in an SLC device, the distribution of \( V_T \) values for programmed and erased cells is shown in Fig. 2 (a). To read an SLC cell, a voltage between the two distributions, called a read reference voltage, is applied to the control gate. If the cell turns on and current flows, the cell is in the erased state, since the applied control gate voltage is higher than the threshold voltage. If the cell does not turn on, the cell is considered to be in the programmed state. By convention, the erased state represents a 1 and the programmed state represents a 0.

It is also possible to represent more than one bit per cell, by detecting the amount of charge on the floating gate rather than just the presence or absence of charge [25]. Multi-level cell (MLC) devices store two bits per cell - a most significant bit (MSB) and a least significant bit (LSB) - and have four discrete \( V_T \) distributions (represented by 00, 01, 10, 11), as shown in Fig. 2 (b). Triple-level cell (TLC) and quad-level cell (QLC) devices are also available. Reading multi-level flash cells requires multiple internal reads at different read
reference voltages to determine which distribution the cell belongs to.

Until recently, NAND devices were planar, meaning all cells were fabricated on a single plane. Devices became more dense as process geometries scaled according to Moore's Law, resulting in smaller, more tightly packed cells. However, this exacerbated the various error mechanisms (described next), leading to higher bit error rates, until a scaling limit was reached. This limitation was overcome with the introduction of 3D NAND, which stacks multiple layers of NAND cells vertically, enabling greater densities with larger, less tightly packed cells [26], [27], [28]. 3D NAND has now been adopted by all NAND manufacturers. With some subtle exceptions (discussed in Section III), the error mechanisms of 3D NAND are broadly similar to those of planar NAND (discussed next). While this research was performed on planar devices, we therefore believe the techniques and findings will be equally applicable to 3D devices.

C. ERROR MECHANISMS

Provided each $V_T$ distribution stays within its allowable read window, as depicted in Fig. 2, no read errors will occur. However, often the distributions can extend into adjacent read windows. In an SLC device, these bits will be in error. In an MLC device, the bits that are read back may be in error, depending on whether they are MSB bits or LSB bits. This is shown in Fig. 3. In this example, some cells from the $00$ distribution are in the $10$ window. The LSB is $0$ in each case, so these bit will not be in error. However, the MSB bits, which should be $0$, will now be read as $1$.

There are a number of reliability mechanisms that cause $V_T$ distributions to move into adjacent read windows. Foremost of these is wear-out [30], [31]. When electrons are tunneled through the oxide layer during programming and erase operations, the oxide is damaged, which traps some of the charge passing through it. This charge accumulates with repeated P-E cycling, which tends to raise the threshold voltage and causes the $V_T$ distributions to move to the right, widen, and overlap further [32], [33]. These charges detrap over time during the retention period, causing the distributions to move back to the left, widen, and overlap further [34], [35]. The shift in $V_T$ distributions is most pronounced at high P-E cycles, when the amount of charge trapped in the oxide is highest. When devices have been cycled to their maximum cycling level, the distributions will be furthest to the right immediately after the last program operation i.e. at zero retention time; and will be furthest to the left after the full retention time has elapsed. Therefore, if the RBER is below the required level at these two extremes, it will be below that level at any retention time in between.

Other error mechanisms include read disturb and program disturb. Read disturb is a phenomenon in which many reads of a cell can add charge to the floating gate and increase its $V_T$, causing the $V_T$ distributions to move into an adjacent read window [15], [16]. However, they are ‘soft’ errors, in that they disappear if the cell is erased and reprogrammed. Program disturb is the result of a program operation disturbing the $V_T$ of an adjacent cell [17], [18], [19], [20]. It is data dependent and can be exacerbated by certain data patterns. To mitigate this, the controller generally randomises the data before storing it to NAND, so these worst-case patterns are avoided [21]. The design of NAND devices can also factor program disturb effects into sequential programming operations to ensure the $V_T$ distributions end up in the desired position [22].

Neither program disturb nor read disturb damage flash cells and, as such, do not determine the end-of-life point of a NAND device. Instead, this is determined by the amount of trapped charge in each cell, as a result of P-E cycling. A key contention of this work is that cells do not trap charge uniformly and that, for a given P-E cycle count, some cells will contain more trapped charge than the rest of the population. We call these weak cells, and postulate that the number of errors in a sector at the end-of-life is correlated with the proportion of weak cells the sector contains. Furthermore, we propose that weak cells have physical properties due to process variations that make them more susceptible to charge trapping, and that these properties are identifiable early in life.

D. NAND DEVICE ARCHITECTURE

NAND devices are comprised of a series of blocks, each of which is subdivided into a series of pages. The devices used for this research have 2,096 blocks, and there are 512 pages per block. Each page has a main area, which can store 16,384 bytes of user data, plus an additional spare area, which can store 1,872 bytes for controller meta data. NAND devices are programmed and read at the page level, but can only be erased at the block level.

Physically, each block consists of a matrix of flash cells, as shown in Fig. 4. Matrix rows are called wordlines (WLS) and matrix columns are called bitlines (BLs).

For MLC devices, each cell stores an MSB and an LSB. A collection of MSBs on a wordline forms an MSB page, and a collection of LSBs on a wordline forms an LSB page. The devices used in this research contain four pages per wordline: MSB and LSB even for even-numbered bitlines, and MSB and LSB odd for odd-numbered bitlines. This is illustrated in Fig. 4, with the MSBs corresponding to the even-
numbered bitlines on WL0 comprising page 4, and the MSBs corresponding to the odd-numbered bitlines comprising page 5 etc. Since there 512 pages per block and four pages per wordline, there are 64 wordlines per block.

NAND devices are designed such that pages are programmed sequentially within a block. This means that LSB pages are programmed first using a coarse programming operation, and the subsequent fine programming of the corresponding MSB page places the $V_T$ distribution in the correct position. Furthermore, it allows manufacturers to take program disturb effects (described in Section II) into account. For example, if even pages on a wordline are always programmed before odd pages, then the even pages can be deliberately under-programmed, in the knowledge that the subsequent odd page programming disturb will shift the $V_T$ distributions closer to the desired position.

### E. FLASH CONTROLLER AND ERROR CORRECTION

On a NAND-based product such as an SSD, NAND devices are managed by a controller chip. As well as containing dedicated hardware, the controller runs intelligent firmware which processes the read and write requests from the host and sends them to the NAND device. It translates these host requests into low-level operations that can be understood by the NAND, and utilises the NAND resources to store data as efficiently as possible. This means that writes to the NAND are spread across the entire device so that no single block wears out prematurely, in a process known as wear leveling.

Another key function of the controller is error correction [39]. When the host issues a write request, the controller splits the data into 1kB or 2kB chunks. Each chunk is sent to the hardware ECC encoder, which generates parity data and appends it to the user data, forming a codeword. The user data portion of the codeword is written to the main area of a page, and the parity data portion is written to the spare area. When the codeword is read back by the controller, it passes through the ECC decoder, which uses the parity data to detect and correct any bit errors in the user data.

The codewords used in this work have 2kB bytes of user data. Because writes are performed at the page level, and there are 16kB of user data per page, eight codewords are written at a time. The portion of a page taken up by a codeword is called a sector, so there are eight sectors per page.

Traditionally, NAND ECC engines have been based on BCH (Bose-Chaudhuri-Hocquenheim) codes. BCH is a relatively simple, well understood ECC code which, for a given codeword size (data bits plus parity bits), provides a fixed level of error correction, such as 100 bit errors per codeword. Increasing the number of bits that can be corrected requires more parity bits to be stored on the NAND, and more decoding logic on the controller. This increases decoding time, which in turn affects read performance. As NAND RBER increased with process scaling, the number of parity bits and the controller area required by BCH grew prohibitively large, and the industry moved towards a more advanced ECC technique called LDPC (Low Density Parity Check).

A major difference between LDPC and BCH is that LDPC does not guarantee that it can correct a fixed number of bit errors per codeword. Instead, it identifies the bits that are most likely to be in error, and tries to guarantee that the fraction of codewords that it fails to decode - known as the codeword error rate (CWER) - is below a certain level. For a given ratio of parity bits to data bits (known as code rate) and a fixed RBER, LDPC has a lower CWER than BCH.

However, this increase in ECC performance is counter-balanced by serious challenges associated with the implementation and operation of LDPC [38]. Firstly, LDPC often requires additional ‘soft’ reads to decode a codeword, which are hugely detrimental to read performance. Secondly, LDPC requires accurate ‘soft information’ associated with the soft reads which can only be acquired via complex mathematical modeling of the NAND device, or large-volume characterisation of raw NAND. Thirdly, the performance of an LDPC implementation cannot be mathematically characterised and requires massive simulation resources, both in terms of the hardware required and the time taken to run the simulations. These significant challenges mean that only the most sophisticated and well-resourced integrators of NAND are capable of implementing an LDPC solution.

### III. RELATED RESEARCH

Previous studies have examined raw bit errors on MLC NAND devices. Cai et al. showed that, on 30-40nm planar MLC devices, RBER increases with P-E cycling and retention time. Furthermore, different page types (MSB even, LSB odd) exhibit different error rates, and error rates are dependent on wordline number, indicating that device architecture plays a significant role in a page’s error characteristics [2]. Subsequent work by the authors on the 19nm devices used for this study supported these findings [3].

Since the position of the $V_T$ distributions change with P-E cycling and retention, a fixed read reference voltage (or set of read reference voltages for MLC devices) between the distri-
buttions will not be optimal at every point of life. Papandreou et al. proposed a technique to track and adaptively change the read reference voltages as the device ages, to minimise RBER [14]. This technique is now commonplace on most SSDs, which periodically perform background read reference calibrations. In this work, reads are not calibrated (default settings used at every point), to demonstrate the feasibility of our approach with a simple controller that does not have the ability to perform, or cannot tolerate the performance overhead associated with, read calibrations.

Luo et al. characterised the error mechanisms of 3D NAND devices [13]. They found that RBER increases linearly with P-E cycles, as per planar NAND. The effect of read disturb and program disturb is less pronounced than in 20nm planar devices, due to the larger process geometries used in the fabrication of 3D devices. They also identified three new error mechanisms that are not observed on planar devices. Firstly, the error rates of each layer in a 3D device are significantly different, due to layer-to-layer process variations. Secondly, there is an early retention loss effect, in which the rate of rate of retention errors is high in a short time period after programming, and slows down over time. Thirdly, retention errors show a dependence on the cell state of surrounding cells. It was shown that all of these mechanisms could be mitigated by tuning the read reference voltage. This study shows the error mechanism principles on which our work is based - wearout, process variations and location dependence - also apply to 3D NAND.

Machine Learning has also been applied to NAND for prediction purposes. Arbuckle et al. cycled 40nm NAND devices to destruction (the point at which program or erase operations could no longer be performed), and used the program and erase times to predict the cycling level at which this occurred. Various machine learning methods were compared, with support vector machines (SVMs) found to give the most accurate results [4].

Previous works by the authors have developed prediction models based on data collected at the rated endurance, to determine how much further each sector could be cycled before exceeding an RBER threshold [3], [5], [6]. Sector errors, program time and erase time were all found to have predictive value for this purpose. Gradient boosting was found to be the most effective machine learning method for this type of data and problem domain.

This paper presents the only study that has proved that a) true end-of-life errors (incorporating both endurance and retention) can be predicted from the very beginning of life, without destructively testing the NAND devices; and b) the prediction system can be tuned, using novel techniques, to provide practical solutions for addressing the high bit error rates of modern NAND devices.

IV. RESEARCH OBJECTIVES AND QUESTIONS

The preceding background discussion informs the motivation for this research. It is known that the majority of sectors are capable of far exceeding the rated endurance, since this is defined by the weakest sectors likely to be encountered. There is currently no means of accurately predicting the true endurance of each sector at the beginning of life. The true endurance of a sector is the highest level the sector can be P-E cycled to, while maintaining the number of errors in the sector below a target (ECC) threshold, assuming the sector can be read at any time during the allowable retention period. This retention constraint can be met by ensuring compliance at the two retention extremes i.e. zero retention (which we call pre-retention), and maximum retention (which we call post-retention).

Such a prediction system would potentially be valuable to manufacturers and integrators of NAND. Firstly, it would enable the use of a less complex and less expensive ECC engine (such as BCH) for current and future generation NAND devices. This is because, rather than the current assumption that failing codewords (to a fixed ECC level) are entirely non-deterministic, these codewords could be preemptively identified and managed before they fail, resulting in a very low CWER. Furthermore, rather than retiring all sectors at the rated endurance to avoid the risk of a small proportion failing, sectors could continue to be cycled and retired as necessary, trading off some capacity for extra endurance.

An important requirement of the prediction system is that it should never predict that a sector will pass at a given cycling level, only for it to fail when it reaches that cycling level. This would produce an uncorrectable error, resulting in data loss. From the predictive model’s point of view, this is a false negative (assuming the model classifies a sector that is predicted to pass as a negative). In machine learning classification, the proportion of true positives that receive a positive prediction is termed sensitivity, so the system has a 100% sensitivity requirement (no false negatives). Predicting a fail for a sector that would actually have passed (i.e. a false positive) is a less serious scenario, simply meaning that the sector could have been cycled further. The proportion of true negatives that receive a negative prediction is called specificity.

A. RESEARCH QUESTIONS

In the development of such a prediction system, this study answers the following research questions:

- **RQ1** What is the achievable performance of the base pre- and post-retention models?
- **RQ2** Can model performance be improved using other machine learning techniques?
- **RQ3a** Can the models be tuned to achieve 100% sensitivity?
- **RQ3b** What is the achievable specificity of each model if 100% sensitivity is achieved?
- **RQ4** What is the endurance versus capacity trade-off if the tuned prediction system was deployed?

1In practice, a very small CWER is tolerable (typically less than 1e-12) so the 100% sensitivity requirement is marginally reduced.
V. EXPERIMENTAL DESIGN

To address the research questions, beginning-of-life and end-of-life data is gathered on raw NAND devices, and predictive models are built from this data. The devices used are planar 19nm MLC devices, which is the smallest process node for planar NAND.

Building the predictive models from the raw data involves casting the problem as a binary classification one. This means the machine learning models output one of two possible values: pass or fail. If the number of errors per sector is less than a predetermined threshold, known as the decision boundary, the sector is deemed a pass; otherwise it is classified as a fail.

As discussed in Section II, each sector consists of 2,048 bytes of user area and 234 bytes of spare area. If all 234 bytes of spare area were used for ECC parity, this would result in a BCH error correction level of 124 bits per codeword, and a low code rate of less than 0.9. In practice, some spare area is needed for other controller meta data. For this reason, an error correction level of 100 bits per codeword was used for this work, which would require only 188 bytes of spare area, and give a higher code rate of 0.92. As well as using less spare area, higher code rates have faster decoding performance since they require less decoding logic. This allows our prediction system to be demonstrated for a practical BCH performance since they require less decoding logic. This allows our prediction system to be demonstrated for a practical BCH implementation. A lower decision boundary also produces more failing sectors, which is beneficial for training the classification models.

The full experimental procedure is summarised as follows:

1) In a data collection phase, perform accelerated endurance and retention testing of raw NAND devices.
   a) After a few cycles (beginning of life), record error and timing information for each sector.
   b) Record sector errors after cycling has completed (pre-retention), and after the retention bake (post-retention).
2) In a data analysis phase, examine the raw test data to identify interesting features, trends or potential early-life predictors.
3) In a machine learning phase, build pre-retention and post-retention classification models that predict if the number of errors in each sector will be above or below the decision boundary.
   a) Collate the data recorded for each sector into a data set.
   b) Partition the data set into several training and test sets.
   c) For each partition, train the evaluate the models.

Each of these steps is described in the following sections.

VI. DATA COLLECTION

The first phase of the research involved testing raw NAND devices in hardware to establish the number of sector errors at the end of life, and to measure various metrics at the beginning of life that may have predictive potential. NAND lifetime is tested in an accelerated manner at elevated temperature. For a given test temperature and an expected use temperature, an acceleration factor for both P-E cycling and retention is calculated using the Arrhenius equation, as set out in the industry standard JEDEC specification [7].

A. TEST PLATFORM

A custom-designed test platform was developed, capable of P-E cycling NAND devices, and measuring both sector errors and program and erase times. The test platform comprises multiple hardware testers controlled by a central server computer. Each hardware tester consists of a Raspberry Pi single-board computer, and a daughter board which houses the NAND device under test (DUT). This is shown in Fig. 5.

![Figure 5: Hardware tester, comprising (a) Raspberry Pi single-board computer and (b) daughter board with socket for NAND chip. The daughter board plugs onto the Raspberry Pi header block, as shown in (c).](image)

The function of the hardware tester is to perform the low-level NAND operations such as erase, program and read. It does this by running software on the Raspberry Pi, which generates the required digital signals via the GPIO pins. The software can also return the number of errors in a sector by comparing the data read from the sector with the data written; and can measure the time taken for read, program and erase operations to complete. It does this by monitoring a NAND pin (R/B) which is brought low for the duration of these operations.

Each tester is connected to the server computer via a series of ethernet switches. The server runs high-level software, which controls the testers in parallel. It manages the test flow and test data, by issuing high-level commands to each tester to invoke the low-level NAND operations, and reading back error and timing data. It also stores this data in a MySQL database on the server.

To enable testing at elevated temperature, the testers are housed in environmental ovens (portable to supply Ethernet and power cables to each tester). The full test platform is depicted in Fig. 6, which shows 12 testers in two ovens for illustrative purposes. In practise, 45 testers in eight ovens were used for this work.

B. TEST PROCESS

In line with the target application and in compliance with [7], blocks were P-E cycled for 500 hours at 81°C. A retention bake was performed for 18 hours at 81°C, which is equivalent to the requirement of three months at 40°C for these enterprise-grade NAND devices. The number of errors per sector were measured both pre- and post-retention, and these reads were performed at 25°C.

Whereas the test outputs sector-level errors after a predetermined cycling level, the goal is to predict each sector’s
achievable cycling level, given a maximum allowable number of errors. Therefore, each block was cycled to one of ten discrete cycling levels. The levels chosen were 6k to 15k cycles inclusive, in steps of 1k cycles. 75 blocks per device were cycled across 45 devices, giving 6,675 blocks in total. There are 4,096 sectors per block, so more than 27 million sectors were tested in total. The number of blocks and sectors cycled to each cycling level is summarised in Table 1.

| Cycling Level | Blocks per Device | Total Blocks | Total Sectors |
|---------------|------------------|--------------|---------------|
| 6,000 Cycles  | 8                | 712          | 2,916,352     |
| 7,000 Cycles  | 8                | 712          | 2,916,352     |
| 8,000 Cycles  | 8                | 712          | 2,916,352     |
| 9,000 Cycles  | 8                | 712          | 2,916,352     |
| 10,000 Cycles | 8                | 712          | 2,916,352     |
| 11,000 Cycles | 7                | 623          | 2,551,808     |
| 12,000 Cycles | 7                | 623          | 2,551,808     |
| 13,000 Cycles | 7                | 623          | 2,551,808     |
| 14,000 Cycles | 7                | 623          | 2,551,808     |
| 15,000 Cycles | 7                | 623          | 2,551,808     |
| Total         | 75               | 6,675        | 27,340,800    |

The following metrics were recorded for each sector after minimal cycling - 6 cycles for 6k blocks, 7 cycles for 7k blocks etc. - to be used as early-life predictors of endurance:

1. Number of sector errors.
2. Page program time.
3. Block erase time.

VII. RAW DATA ANALYSIS

A. END-OF-LIFE SECTOR ERRORS

The empirical cumulative distribution function (ECDF) and complementary cumulative distribution function (CCDF) of pre- and post-retention sector-level errors at each cycling level are shown in Fig. 7.

Fig. 7 (a) and (b) show that the vast majority of sectors have low errors, even after 15k cycles, with a small minority of high-error sectors producing a long distribution tail. At a decision boundary of 100 bits, only 3,440 sectors of the 27 million tested (0.013%) failed pre-retention. Post-retention, only 1,272 sectors failed (0.005%). Mostly, the sectors that fail post-retention also fail pre-retention, as shown in Fig. 8.

Fig. 7 (c) and (d) highlight the mean of each cycling level by zooming in on the body of the distributions. As expected, the mean of each distribution increases with cycling level. Furthermore, the post-retention means are higher than the pre-retention means. This is also expected, as RBER tends to increase with retention time due to distribution widening. However, the finding above that fewer sectors fail post-retention than pre-retention, would seem to contradict this. We believe that there are two reasons for this.

Firstly, we postulate that a) these failing sectors contain a high proportion of weak cells, and b) cycling causes a large $V_T$ shift to the right in weak cells, pushing them outside their allowable read window. The leftward $V_T$ shift during retention means many of these cells have moved back into their allowable read window at the post-retention point. Secondly, the effect is noticeable because we use the same read levels for both the pre- and post-retention measurements. If calibrated read levels (discussed in Section III) were used for every read, the number of failing sectors would decrease, both pre- and post-retention. Furthermore, we expect that there would be fewer failing sectors pre-retention than post-retention, as the overlap between distributions is highest at the post-retention point, resulting in the highest RBER.

Fig. 7 (e) and (f) show the complementary cumulative distribution function, which highlights the tail of each distribution. It shows that no 6k or 7k sectors fail at an ECC level of 100 bits (either pre- or post-retention), with the first fails occurring at 8k. This indicates that the endurance of these devices when using a 100-bit ECC level is 7k P-E cycles. However, even after 15k cycles, the failure rate is less than $1e^{-3}$, reinforcing the finding that the true endurance of most sectors is well in excess of 7k cycles. The plots also reveal a lot of crossover between cycling levels in the tail region.

This is an important finding: CWER is generally assumed to be a strong function of mean RBER [9], but these results show that CWER is actually determined by the RBER distribution tails rather than the mean RBER. This is illustrated by the fact that the 15k cycling level has the highest mean RBER but the 8k cycling level has a higher tail, both pre- and post-retention. This finding is consistent with the results of a large-scale study of datacenter SSD failure rates, which found that CWER is not correlated with mean RBER [10].

B. SECTOR ERRORS BY LOCATION

1) Wordline Dependence

To investigate the influence of location within a block on sector errors, the average pre- and post-retention errors per sector was analysed for each page type - LSB even, LSB odd, MSB even, MSB odd - across all 128 wordlines. To remove the effect of cycling level on sector errors, only sectors cycled to a single cycling level (6k) are considered. Results are shown in Fig. 9.
It is clear from Fig. 9 that each of the four page types per wordline exhibit different raw bit error rates. LSB even pages have the highest RBER, followed by MSB even, then LSB odd, and finally MSB odd. Even pages have higher RBER than odd pages because, as discussed in Section II, odd pages on a wordline are programmed after the corresponding even pages, which disturbs the even-page data.

Furthermore, LSB pages have higher RBER than the corresponding MSB pages. This is the result of a chip design decision which determines the placement of each $V_T$ distri-
FIGURE 8. Venn diagram showing the number of sectors that fail pre- and post-retention. 1,219 sectors fail both pre and post, with only 53 sectors failing post but not pre.

The finding that the worst pages are pages 1 and 7 is not surprising, as these are the LSB even\(^2\) pages on the first two wordlines, which Fig. 9 (a) and (b) showed have the highest average errors per sector. However, a closer inspection of Fig. 9 (c) and (d) reveals that seven of the ten worst pages pre-retention are LSB even pages, with MSB even pages accounting for the remaining three. The trend is reversed post-retention, with the worst 10 pages comprising seven MSB even and three LSB even. The fact that such a small subset of pages account for most NAND codeword failures is a crucial finding that has never before been reported.

The ten pages with the most failing sectors are shown in Fig. 9 (c) and (d). The last bar in red consists of all other pages that failed. Of the 512 page numbers, only 101 ever produced failing sectors pre-retention, and this number reduces to 34 post-retention. Furthermore, the vast majority of failing sectors come from a very small percentage of page numbers. For example, page 1 accounts for 57% of all failing sectors pre-retention, and 79% of all failing sectors post-retention. Page 7 accounts for 19% of failing sectors pre-retention, and 5% post-retention. The fact that such a small subset of pages account for most NAND codeword failures is a crucial finding that has never before been reported.

Cycling and the width of each \(V_T\) read window, as discussed in Section II. These chip parameters are chosen to minimise overall RBER and will vary from one NAND manufacturer and NAND family to another.

Fig. 9 (a) and (b) also show a clear RBER dependence on wordline number. In general, RBER is highest for low-numbered wordlines, particularly for LSB even pages. There are also two points - around wordlines 20 and 70 - at which the RBER changes sharply. This wordline dependence is likely due to wafer fabrication processing effects.

This analysis shows that some pages are more likely to exhibit higher RBER than others, meaning that page number may be an important input to any prediction model.

2) Failing Pages

The ten pages with the most failing sectors are shown in Fig. 9 (c) and (d). The last bar in red consists of all other pages that failed. Of the 512 page numbers, only 101 ever produced failing sectors pre-retention, and this number reduces to 34 post-retention. Furthermore, the vast majority of failing sectors come from a very small percentage of page numbers. For example, page 1 accounts for 57% of all failing sectors pre-retention, and 79% of all failing sectors post-retention. Page 7 accounts for 19% of failing sectors pre-retention, and 5% post-retention. The fact that such a small subset of pages account for most NAND codeword failures is a crucial finding that has never before been reported.

The finding that the worst pages are pages 1 and 7 is not surprising, as these are the LSB even\(^2\) pages on the first two wordlines, which Fig. 9 (a) and (b) showed have the highest average errors per sector. However, a closer inspection of Fig. 9 (c) and (d) reveals that seven of the ten worst pages pre-retention are LSB even pages, with MSB even pages accounting for the remaining three. The trend is reversed post-retention, with the worst 10 pages comprising seven MSB even and three LSB even. The fact that MSB even pages account for such a high proportion of failing sectors is very surprising, as Fig. 9 shows that, for every wordline, MSB even pages have lower average errors than LSB even, both pre- and post-retention. This shows that mean page-level RBER is not a good indicator of pages that are likely to fail.

VIII. MACHINE LEARNING

Once the raw data was gathered, the next phase was to use machine learning to predict the number of end-of-life errors per sector from the early-life data. Two machine learning models were built: one for pre-retention and one for post-retention. The predicted endurance of a sector is the lowest cycling level predicted by both models.

A. MACHINE LEARNING METHOD

Gradient boosting was chosen as the machine learning method, as prior work by the authors showed it to be the most effective method on this type of data [5]. Gradient boosting is a machine learning technique that trains an initial weak model (usually based on decision trees), and then adds successive stages of models to minimise an error or loss function using a gradient descent-type procedure [8]. The \texttt{gbm} library in \texttt{R} was used for the implementation. Internal parameters were first tuned by grid search.

B. MODEL INPUTS AND OUTPUTS

The data recorded for each sector, along with the sector’s page number and the cycling level it was tested to, was collated into a data set. Each data point in the set comprises the attributes outlined in Table 2.

| Attribute | Details |
|-----------|---------|
| Input 1   | Cycling level |
| Input 2   | Page number  |
| Input 3   | Early-life program time |
| Input 4   | Early-life erase time |
| Input 5   | Early-life sector errors |
| Output    | 0 if sector errors after cycling <= 100 |
|           | 1 if sector errors after cycling > 100 |

Cycling level is the base input to each model. When supplied to the model, along with the other four inputs, the model will predict if the sector will pass (output 0) or fail (output 1) to that cycling level.

C. MODEL TRAINING AND TESTING

Typically building and validating machine learning models involves partitioning the data set into a training set and a test set. Training a model on a heavily imbalanced data set such as this (more than 99.9% of sectors pass, both pre- and post-retention) is problematic because the model may simply ignore the minority class and predict that all sectors will pass, achieving greater than 99.9% accuracy. A solution is to undersample the majority class so that both classes are of a similar size [11].

Therefore, all failing sectors were first removed from the data set, and these were divided into eight equal groups. One of these groups was removed, which, in conjunction with randomly chosen passing samples, formed the test set. A

\(^2\)Due to a chip design feature, pages 1 and 7 are regarded as even pages.
10:1 pass:fail ratio was chosen for the test set, to increase the probability of choosing passing samples close to the 100-bit decision boundary, where the model is likely to struggle to make correct predictions. The remaining seven failing-set groups, in conjunction with randomly chosen passing samples, formed the training set. A 1:1 pass:fail ratio was used for the training set to ensure balanced classes.

This training/testing procedure was repeated 30 times, with a different partitioning of the failing set each time, and therefore a different test set. Results are averaged across all 30 runs. The procedure is similar to standard $k$-fold cross-validation, which splits the data into $k$ groups and performs $k$ machine learning runs, using a different group for testing on each run. The reason for our modification is to facilitate the investigation of oversampling techniques, such as the duplication or synthesis of failing samples. Generating a data set containing duplicate or synthetic samples, and then removing a test set, violates the assumption of independence of the test set, since it may contain samples (either direct copies or derived) from the training set.

### D. MODEL EVALUATION

Three metrics are used to evaluate model performance: sensitivity ($Sn$), specificity ($Sp$) and area under the receiver operating characteristic curve (ROC AUC). As discussed in Section IV, sensitivity and specificity are a measure of misclassified failing samples (false negatives) and passing samples (false positives) respectively.

Machine learning models output the probability (between 0.0 and 1.0) that the sample is a negative (class 0) or a positive (class 1), and classification models convert this probability to a class. Probabilities below 0.5 are converted to class 0 and
those above are converted to class 1. Probabilities close to 0 or 1 indicate the model is very confident in its prediction, whereas probabilities close to 0.5 indicate less confidence. ROC AUC is a metric that shows how likely the model is to assign a higher prediction probability to a randomly chosen positive sample than to a randomly chosen negative sample. It is found by sorting the test samples by prediction probability in ascending order. For an ideal classifier, the list will contain all the negative samples followed by all the positive samples, without any mixing of the two, resulting in an AUC of 1.0. Lower AUC scores indicate that the model assigns higher prediction probabilities to some negative samples than to some positive samples.

**IX. INITIAL PREDICTION MODEL**

The first machine learning experiments trained and tested models using the procedure outlined in Section VIII. To investigate the impact of different input attributes on model performance, four sets of models were trialed. The first set of models used cycling level, page number and early-life sector errors as inputs. In addition to these three, the second set of models used program time; the third set used erase time; and the fourth set used both program and erase time. Results are presented in Table 3.

**TABLE 3.** Sensitivity (Sn), Specificity (Sp) and AUC of models that use different sets of input attributes. Attributes shown are in addition to page and cycling levels.

| Input Attributes                  | Pre-Retention | Post-Retention |
|----------------------------------|---------------|----------------|
|                                  | Sp  | Sn  | AUC  | Sp  | Sn  | AUC  |
| sector errors                    | 97.52 | 97.62 | 0.9965 | 98.84 | 99.41 | 0.9992 |
| sector errors program time       | 98.09 | 98.62 | 0.9981 | 99.01 | 99.63 | 0.9992 |
| sector errors erase time         | 98.55 | 99.17 | 0.9990 | 99.02 | 99.69 | 0.9995 |
| sector errors program time erase time | 98.77 | 99.35 | 0.9990 | 99.00 | 99.66 | 0.9994 |

Table 3 highlights a number of notable points. Firstly, it shows that it is possible to make both pre- and post-retention predictions from measurements taken at the beginning of life with remarkable accuracy. Specificity in excess of 98.7%, sensitivity in excess of 99.35% and AUC in excess of 0.999 are achievable. Secondly, for each input attribute grouping, post-retention results are higher than pre-retention results across all three evaluation metrics. This indicates that pre-retention is a more difficult prediction challenge.

Thirdly, most of the predictive power comes from a combination of page number and early-life sector errors. This is not surprising, since Section VII showed that end-of-life sector errors are highly dependent on page number. A sector with high early-life errors indicates a higher number of weak cells, which should produce higher errors at end of life.

Fourthly, the predictive power of the model is improved with the addition of program time and/or erase time, with erase time providing the most benefit. Since program and erase time are effectively a measure of how difficult it is to program or erase cells in a page or block, we postulate that this provides a measure of cell quality, which in turn affects end-of-life error rate. For pre-retention models, optimal performance is achieved when both program and erase time are used. For post-retention models, erase time alone achieves slightly higher performance. Therefore for the remainder of this work, pre-retention models will use both program and erase time, and post-retention models will use erase time only.

**X. MODEL IMPROVEMENT**

An analysis of the incorrect predictions from this initial model revealed that most misclassifications were due to a shortage of training data in certain areas of the search space. To increase the volume of training data, various oversampling techniques were investigated. Oversampling involves increasing the minority class through duplication of samples or generation of synthetic samples. Both techniques were examined here.

Oversampling by duplication involves duplicating each failing sample in the training set a fixed number of times, called the oversampling factor, and randomly selecting the same number of passing samples to balance both classes. A number of oversampling factors were investigated, ranging from 10 to 200.

Synthetic samples were generated using a variation of the synthetic minority over-sampling technique (SMOTE), which creates synthetic samples by randomly combining attributes from similar samples in the minority class [40]. The procedure for generating the input attributes of each synthetic sample is detailed in Table 4.

**TABLE 4.** Procedure for generating the attributes of each synthetic sample.

| Attribute                  | Procedure                                      |
|----------------------------|------------------------------------------------|
| Page Number                | Page number of the two parent samples          |
| Cycling Level              | Randomly choose valid cycling level between the cycling levels of the two parent samples |
| Sector Errors              | Randomly choose an integer number of errors between the errors of the two parent samples |
| Erase Time                 | Randomly choose erase time of one of the parent samples |
| Program Time               | Randomly choose program time of one of the parent samples |

**A. RESULTS AND DISCUSSION**

The results for both synthetic-based oversampling and duplication-based oversampling are shown in Fig. 10. The results for standard sampling (as per Section IX) are also shown for comparison. Duplication results are presented for oversampling factors of both 10 (Dup10) and 20 (Dup20). For simplicity, only pre-retention results are shown.

Fig. 10 shows that, in contrast to duplication, synthetic-based oversampling is the only technique that outperforms standard sampling across all three evaluation criteria, producing specificity of 98.95%, sensitivity of 99.61% and AUC of 0.9993.
XI. MODEL TUNING

As outlined in the research objectives, a desirable feature of the model is that it could be tuned to provide 100% sensitivity. A common technique to trade off specificity for sensitivity is to reduce the pass/fail threshold on the test data, from the default probability of 0.5 [37]. However, Fig. 11 shows that such an approach would not eliminate the misclassification of failing samples, since their prediction probabilities extend down to zero in a relatively linear fashion, for all sampling techniques. Achieving 100% sensitivity would require the threshold to be lowered to zero, which would simply predict all samples as a fail, rendering the model useless.

To overcome this problem, an alternative threshold-based approach was investigated. This involved reducing the 100-bit decision boundary for model training, while maintaining the 100-bit boundary for model testing. A lower decision boundary ensures more failing samples in the training data (for example, more samples fail to 90 bits than to 100 bits), and biases the model towards the failing class (for example, if the model is trained to classify a 90-bit sample as a fail, it should classify a 100-bit test sample as a fail with a higher degree of confidence).

Four different training-set decision boundaries were trialled: 90-bit, 80-bit, 70-bit, and 60-bit. The number of machine learning runs was increased from 30 to 150 to improve the statistical significance of the test.

A. RESULTS AND DISCUSSION

The 80-bit test was found to produce best sensitivity, yielding no misclassified failing samples post-retention and just one misclassified failing sample pre-retention (from 64,500 failing samples tested in total). This sample had a prediction probability of 0.489, indicating it was a marginal misclassification.

The full distribution of pre- and post-retention failing sample prediction probabilities is shown in Fig. 12. The y-axis is truncated in each plot - the upper bin actually extends to approximately 64k samples. This shows that, for both pre- and post-retention, the vast majority of failing samples are correctly classified with a high degree of confidence, with a small tail extending down to the pass/fail threshold of 0.5. This contrasts with the original distribution shown in Fig. 11, which had uniformly distributed prediction probabilities all the way down to zero.

It is clear from Fig. 12 that a pass/fail threshold of less than 0.489 would prevent any failing samples in our data from being misclassified, thereby achieving 100% sensitivity. To add a guardband for unseen data, the threshold can be reduced further, at the expense of degraded specificity. Table 5 summarises the results for both the default pass/fail threshold of 0.5, and for a reduced threshold of 0.45. The threshold reduction results in a 0.3% and 0.2% reduction in specificity for pre- and post-retention respectively. It should be noted that threshold tuning has no effect on AUC. A threshold of 0.45 provides ample guardband and is used as the pass/fail threshold for the final prediction system, described next.

| Pass/Fail Threshold | Pre-Retention | Post-Retention |
|---------------------|---------------|----------------|
|                     | Sp  | Sn  | AUC   | Sp  | Sn  | AUC   |
| 0.5 (default)       | 97.5 | 99.9993 | 0.9993 | 98.3 | 100  | 0.9996 |
| 0.45                | 97.2 | 100   | 0.9993 | 98.1 | 100  | 0.9996 |
XII. FINAL PREDICTION SYSTEM

The predicted endurance of each sector is the maximum cycling level that both pre- and post-retention models predict will pass. This section examines the predicted endurance of each sector by using the same test set for pre- and post-retention models (for each of the 30 runs). Since only 0.03% of all samples fail across all cycling levels pre-retention, the same ratio of passing and failing samples is used for the test set, which comprises 150 failing samples and 450,000 passing samples. The number of sectors predicted to pass is then analysed, and this is compared to the number of sectors that actually pass at each cycling level.

A. RESULTS AND DISCUSSION

Fig. 13 shows the percentage of predicted and actual sectors that pass both models, as a function of cycling level. This plot effectively shows the reduction in storage capacity as endurance is extended.

![Capacity vs Endurance](image)

**Fig. 13.** Storage capacity versus endurance.

In terms of actual endurance, no sector fails up to 7k cycles (to a 100-bit ECC threshold), and the device can operate at 100% capacity. Some sectors would fail beyond this point, and these could be preemptively retired, resulting in a capacity reduction. However, at 10k cycles, 99.99% of sectors still pass. Even at 15k (more than twice the actual endurance of the worst-case sectors), 99.89% of sectors pass.

The requirement for 100% sensitivity means that some sectors that would have passed to a given cycling level will be predicted to fail, meaning that the predicted capacity is lower than the actual capacity. The divergence between predicted and actual capacity increases with cycling level, as the model is more likely to predict that higher cycling levels will fail. However, at 10k cycles, 98.8% of sectors are (correctly) predicted to pass, and at 15k cycles 86.1% are predicted to pass.

The opportunities afforded by this prediction system are clear. By performing a simple, non-destructive test at the beginning of a NAND device’s life, the true endurance of each sector can be predicted. This may enable simpler, more cost-effective ECC schemes such as BCH to be used on current-generation NAND devices. Such a deterministic ECC scheme would allow most sectors to be cycled far beyond their rated endurance, without risking data loss. This work has been based on a fixed error correction capability of 100 bits per sector, but models can be trained for any target ECC level.

An alternative to retiring sectors, which reduces chip capacity, may be to use enhanced ECC schemes for worst-case sectors. This could allow all sectors to be cycled to similar levels, enabling an endurance extension for the whole device. The principle of having ECC engines which can be adapted for different error rates has previously been proposed [36].

XIII. CONCLUSIONS

This paper has shown that there is a large variation in the endurance of NAND flash sectors, due to a combination of processing effects and device architecture. We show, for the first time, that sector RBER after different cycling levels can be predicted with remarkable accuracy, using a combination of sector address and measurements taken at the beginning of life. Each of the research questions posed in Section IV have been answered.

If only page number and beginning-of-life sector errors are used as inputs to the model, sensitivity and specificity in excess of 97.5%, and AUC in excess of 0.9965, are achievable for both pre- and post-retention models. Adding program time and/or erase time improves sensitivity to more than 99.3%, specificity to more than 98.7%, and AUC to more than 0.9990 (RQ1a and RQ1b). The performance of the models can be improved using oversampling techniques. A custom variation of the SMOTE approach yielded best results, improving sensitivity to more than 99.6%, specificity to more than 98.9%, and AUC to more than 0.9993 (RQ2).

The models can be tuned to achieve 100% sensitivity using a two-step process. Firstly, models are trained to a lower decision boundary of 80 bits. Secondly, models are tested to the required decision boundary of 100 bits, and the pass/fail prediction probability threshold is reduced (RQ3a). This is a desirable feature since it ensures that sectors that would fail at a given cycling level are not incorrectly predicted to pass. Using a conservative pass/fail threshold of 0.45 yields a specificity of 97.2% (RQ3b).

Using a combination of pre- and post-retention tuned models, a deployable prediction system can be realised which trades capacity for endurance. At 1.4x the endurance of the worst sector, 98.8% of sectors are still correctly predicted to pass, while at 2.1x, 86.1% are correctly predicted to pass (RQ4).

The potential of this technique has been demonstrated on 19nm planar NAND devices. Future work will consider 3D NAND devices, and will examine the trade-off between endurance, retention and ECC level.
REFERENCES

[1] A. Norwood, J. Erensen, J. Unsworth, B. Ray, A. Priestly, and G. Brocklehurst, “Market share analysis: semiconductors, worldwide, 2020,” Gartner Inc., Stamford, CT, USA, Tech. Rep. 000742738, 2021.

[2] J.-D. Park, J. Jeong, S. Lee, Y. Song, and J. Kim, “Improving endurance and error reduction in NAND flash using machine learning,” in Proc. NVMFS, Aachen, Germany, 2017, pp. 1-8.

[3] A. Norwood, J. Erensen, J. Unsworth, B. Ray, A. Priestly, and G. Brocklehurst, “Market share analysis: semiconductors, worldwide, 2020,” Gartner Inc., Stamford, CT, USA, Tech. Rep. 000742738, 2021.
BARRY FITZGERALD received the B.Eng. degree in electrical engineering from University College Cork, Ireland, in 1998. He is currently pursuing the Ph.D. degree in electronic engineering from Limerick Institute of Technology, Ireland.

From 1998 to 2010 he was a Test Development Engineer and Senior Design Evaluation Engineer at Analog Devices. From 2014 to 2017, he was Engineering Manager for flash-based startup company NVMdurance. Since 2019, he has been an Assistant Lecturer at Limerick Institute of Technology. His research interests include flash memory, data storage and applications of machine learning.

CONOR RYAN received the B.A. and Ph.D. degrees in computer science from University College Cork, Ireland, in 1992 and 1996 respectively.

He is Professor of Machine Learning in the Department of Computer Science and Information Systems at the University of Limerick and Director of the Biocomputing-Developmental Systems group, Limerick, Ireland. He is a Fulbright Scholar and worked in the Computer Science and Artificial Intelligence Laboratory (CSAIL) in Massachusetts Institute of Technology in 2013. His research interests are in genetic programming, genetic algorithms and grammatical evolution, and he has been a Science Foundation of Ireland funded Investigator since 2002. He has published over 250 papers in core computer science areas such as evolutionary computation, flash memory, and automatic program generation, as well as the application of machine learning to domains as varied as mammography, autoparallelization, stock market trading, aphasia (speech therapy), Voice over Internet Protocol (VoIP) and robotics, and holds several granted patents in electronics.

JOE SULLIVAN received the B.Eng. degree in electronic engineering from University of Brighton, England, in 1994 and the Ph.D. degree in computer science from University of Limerick, Ireland, in 2013.

He has extensive industrial experience and has held engineering roles at Ashling Microsystems and Analog Devices. Since 2001 he has been a Lecturer in the Electronic Engineering Department, Limerick Institute of Technology, Ireland.

His research interests include flash memory, data storage, and applications of machine learning and evolutionary computation, including control parameter optimisation and automatic program generation. He was co-founder and CTO of NVMdurance, a company dedicated to extending the life of NAND flash memory. He holds nine granted US patents and has numerous published papers in computer science and electronics.