# Evolving a Retention Period Classifier for use with Flash Memory

Damien Hogan[1], Tom Arbuckle[1], Conor Ryan[1] and Joe Sullivan[2]

[1]*Computer Science and Information Systems, University of Limerick, Limerick, Ireland*

[2]*Electrical and Electronic Engineering, Limerick Institute of Technology, Limerick, Ireland*

Keywords: Genetic Programming, Binary Classifier, Solid State Drive, Flash Memory.

Abstract: Flash memory based Solid State Drives (SSDs) are gaining momentum toward replacing traditional Hard Disk Drives (HDDs) in computers and are now also generating commercial interest from enterprise data storage companies. However, storage locations in Flash memory devices degrade through repeated programming and erasing. As the storage blocks within a Flash device deteriorate through use, their ability to retain data while powered off over long periods also diminishes.

Currently there is no way to predict whether a block will successfully retain data for a specified period of time while powered off. We detail our use of Genetic Programming (GP) to evolve a binary classifier which predicts whether blocks within a Flash memory device will still satisfactorily retain data after prolonged use, saving considerable amounts of testing time. This is the first time a solution to this problem has been proposed and results show an average of over 85% correct classification on previously unseen data.

## 1 INTRODUCTION

Solid State Drives (SSDs) (Chen et al., 2009) store data electronically using solid state memory, and are gaining momentum towards eventually replacing the traditional Hard Disk Drives (HDDs) used to store data in computers. Since SSDs are based on solid state memory (normally Flash memory), they do not contain any moving parts and are faster, lighter, generate less noise, and emit less heat than their electromechanical counterparts.

The strengths of SSDs are all due to the NAND Flash memory (Pavan et al., 1997) upon which they are based. Advantages of Flash include low power consumption, non-volatility, speed, small size, low heat emission, and durability. The two main weaknesses of Flash memory are known as endurance and retention. These are concerned with memory *blocks*, the smallest *erasable*[1] unit in Flash, and refer to their finite lifetime (measured in number of writes) and ability to retain contents without power.

The bit error rate (BER) (Mielke et al., 2008) is calculated by counting the number of incorrect bits when the data written to a block is compared to the data read back from the same block. A single error occurs when the data bit stored at a location changes value. Error Correction Codes (ECC) (Yaakobi et al., 2010) identify and correct these errors, with the devices tested as part of this research capable of correcting 12 bits per 528 bytes. Blocks are marked as bad and removed from service when the BER exceeds the maximum number of bits correctable through ECC.

Over time, through repeated writes, generally referred to as programming and erasing (p/e cycling), blocks degrade and the BER increases until the accumulating errors can no longer be corrected through ECC and the location becomes unreliable. This is known as the *endurance* of the device and is quantified by manufacturers as the number of p/e cycles each block can reliably complete.

Although Flash memory is non-volatile, it is not perfect, and the contents of storage locations will slowly degrade if left powered off, i.e. the charge stored can leak, making it increasingly more difficult to establish whether a location contains a 1 or 0. The longer a device is left powered off, the higher the BER will be. It is possible for a device to be left without power for too long, resulting in the BER increasing above the ECC level. The *retention* of a device specifies the length of time for which a block can reliably store data.

A trade-off exists between the endurance and retention characteristics of Flash memory devices meaning that devices with higher endurance will have

[1]As described in Section 2.1, there are different sized regions which can be accessed, depending on the operation.



Hogan D., Arbuckle T., Ryan C. and Sullivan J..
Evolving a Retention Period Classifier for use with Flash Memory.
DOI: 10.5220/0004116200240033
In *Proceedings of the 4th International Joint Conference on Computational Intelligence* (ECTA-2012), pages 24-33
ISBN: 978-989-8565-33-4
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

lower retention and vice versa. Devices must meet both the endurance and retention specifications outlined by the manufacturer, but tests to evaluate actual endurance and retention are hugely time consuming. It is important to test the worst-case retention of a device – i.e. the retention when blocks have already been significantly degraded through p/e cycling, since the device must meet the specified retention requirement regardless of the number of cycles completed.

To examine the retention period of a block within a Flash device, the block is first cycled at high temperature (Mielke et al., 2006) to simulate real-world, lifetime usage. Following this, a predefined data pattern is written to the device before it is placed in a high temperature oven[2]. After a fixed period of time the previously stored data is then read from the test block and the corresponding BER is calculated, allowing evaluation of the retention ability of the device. In total, this test process takes almost eight days.

There is currently no method to predict the deterioration of data stored in a Flash memory block over a fixed period of time. In this paper, we use Genetic Programming (GP) (Koza, 1992; Poli et al., 2008) to evolve a *Retention Period Classifier* which predicts whether Flash memory blocks will retain data for the required length of time. This binary classifier determines whether the number of errors following the retention period will exceed a predetermined decision boundary, allowing us to determine quickly blocks which will not correctly retain data over the course of a retention period without having to complete the time consuming assessment itself.

Analysis of the data points acquired from destructive tests on actual Flash devices shows the difficulty of the problem while the results of classification tests performed on previously unseen data show its potential for real world application. We show that, in spite of this analysis revealing inconsistencies and even contradictions in the data, we evolve a classifier that can correctly predict the result of a retention period assessment over 85% of the time, across a variety of cycling conditions on previously unseen data.

The remainder of this paper is structured as follows. Section 2 will provide more information on SSDs and Flash memory while Section 3 introduces research related to this topic. Section 4 gives details of the experiments used to gather the retention data and also our GP configuration while Section 5 presents our results. We then give an overview of our proposed future work in Section 6 before concluding the paper in Section 7.

[2]Arrhenius' Equation is the standard technique used to determine the variable values (duration and temperature) for temperature accelerated tests.

# 2 BACKGROUND

## 2.1 Flash Memory

Flash memory is a non-volatile, electrically erasable programmable read only memory (EEPROM). Non-volatility is achieved through a floating gate transistor (Kahng and Sze, 1967) which utilises an insulating oxide layer in order to maintain the floating gate's charge. The quantity of charge stored on the gate is read to determine the data stored at that location.

There are two main forms (Brewer and Gill, 2008) of Flash memory that are named after the organisation of the arrays of floating gates within the devices, namely, NOR Flash and NAND Flash, which operate in different ways, making them suitable for different applications. NOR Flash cells are connected in parallel which allows the cells to be read and programmed individually while the NAND Flash memory configuration is based on cells in series. NOR Flash is ideal for code storage and execution applications while the NAND Flash memory configuration makes it denser and therefore cheaper than NOR memory. NAND Flash memory is used in data storage applications and is the standard form of memory used in SSDs.

The traditional method of storing data in Flash cells is to store a single bit per floating gate transistor. This is the fastest and most reliable method and is referred to as single level cell (SLC) Flash. In order to increase the density of cells and decrease the costs of Flash, multilevel cell (MLC) technology has emerged which lowers the cost of Flash devices by storing multiple bits per floating gate.

MLC Flash SSDs are more appealing to consumers due to their lower cost and higher capacity but do not offer the same performance (in terms of endurance, speed or retention) as SLC based drives. However, Flash manufacturers are focusing more on the consumer MLC market, since these devices are approximately half the cost of SLC devices to produce and their performance in general is good enough for the consumer market, e.g. mobile devices, memory sticks, and so on. However, this has led to a significant reduction in the number of SLC devices being manufactured, effectively raising the price of SLC, with SLC devices selling for up to 6 times the price of MLC devices (Shread, 2009).

NAND Flash memory is composed of accessible regions known as pages (comprised of arrays of cells) and blocks with the size of these areas varying between devices due to factors such as storage capacity. In the MLC devices tested as part of this research, a page contains 4096 bytes for storage and an additional 224 spare bytes (not visible to the user) which are typ-

ically used for error correction and other meta-data, while a block is composed of 128 pages with 16384 blocks on the chip. Read and program operations can occur at page level with a block being the smallest area that can be erased. An important characteristic of programming Flash memory is that locations must be erased before they can be programmed.

Unlike NOR Flash, which is required to be error-free, ECC is essential for NAND Flash memory, working at the page level and correcting a small number – for example 12 bits per 528 bytes – of single bit errors per page. NAND chips may also ship with some bad blocks which are identified during the manufacturing process. These blocks are recorded on the device and not used, allowing far more chips to be shipped, lowering chip cost.

#### 2.1.1 Endurance versus Retention

One of the main characteristics of Flash memory is the degradation (Aritome et al., 1993) through repeated p/e cycling of the oxide layer insulating each floating gate. This leads to blocks having a finite lifetime, termed *endurance*, generally quantified by manufacturers as a maximum number of p/e cycles before the block will become unreliable or fail completely.

A second important characteristic of Flash memory, known as *retention*, is the ability of stored bits within the device to retain their state over long periods of time. Retention errors occur due to leakage of electrons (Aritome et al., 1990) from a cell over time and can be accelerated by increased temperature and wear (through p/e cycling).

A typical manufacturer's specification for SLC NAND Flash endurance is 100,000 cycles, while MLC endurance is generally far less and usually in the region of 5,000 to 10,000 cycles. Specified retention for MLC NAND consumer grade devices is typically 10 years while the retention of enterprise devices ranges from 3 months up to 1 year. The difference between the retention rating of consumer and enterprise devices is due to the endurance / retention trade-off as enterprise devices accept a lower retention rating in order to gain a higher endurance specification.

#### 2.1.2 Operating Parameters

Flash memory devices contain a number of control registers, which store the various operating parameters required by the device including values to represent voltage levels and timings for read, program, and erase operations. Many of these values are interdependent and all are set before the device leaves the manufacturing plant, remaining unchanged for the lifetime of the device.

### 2.2 Solid State Drives

SSDs are designed to mirror HDDs in terms of their external attributes such as size, form factor and communication interface. Since SSDs do not contain any moving parts, their latency (the delay between requesting and receiving data) is typically orders of magnitude better than HDDs and they also perform much faster, due to their large amount of internal parallelism, with many Flash components providing data at the same time. SSDs use far less energy than HDDs, which leads to longer battery life in mobile devices and, in enterprise situations, can lead to huge savings on cooling expenditure since using less power generates less heat.

Momentum is now growing towards using SSDs in enterprise environments for large scale data storage. However, consumer grade SSDs favour cost over performance, so MLC NAND Flash memory is the standard memory for them, while the performance demands of an enterprise data storage environment require that the more expensive SLC NAND Flash memory is used. Enterprise data storage companies are hopeful that the improvement of MLC NAND Flash performance will continue so that it will soon be capable of meeting the demands of an enterprise storage device.

## 3 RELATED RESEARCH

(Sullivan and Ryan, 2011) reported on their application of an Evolutionary Algorithm (EA) to the problem of Flash memory degradation. Their research investigated the viability of developing a hardware platform to facilitate the use of an EA to automatically discover improved operating parameter settings within NOR Flash memory. The results of their experiments showed an average endurance improvement of between 250% and 350% with a maximum achieved improvement of 700%.

Information is not available as to how manufacturers set these operating parameters, but Sullivan and Ryan's research has shown that they are not optimal. This is in agreement with our view that a GA can be used to adjust them in order to improve the endurance of MLC NAND Flash memory and will be discussed further in Section 6.

Research by (Grupp et al., 2009) determined that the performance of Flash memory varies significantly between devices. Their results showed that program operations on certain pages within blocks are faster than others. These pages perform approximately 5 times faster than the rest of the pages. MLC NAND

program performance was found to have increased on average by 10-15% over the lifetime of each block, indicating that, as the devices degrade, it becomes easier to program them. This is due to the fact that the insulating oxide layer has deteriorated, resulting in less resistance to the tunnelling of electrons through it.

The work by Grupp et al. found that even devices made by the same manufacturer can perform at different levels. This reinforces our belief that tests must be performed on a number of devices, and in particular, on a number of Flash memory devices of the same specification and model, in order to generate results which can be generalised to all devices of that type.

(Desnoyers, 2010) found that the actual lifetimes of blocks in a Flash memory device vary greatly with some lasting up to 100 times longer than the manufacturers' specifications. Endurance was tested by repeatedly p/e cycling a single page within a block with all zeros. Desnoyers also reported that program time decreases with block usage, while erase time increases as the block wears.

Tests performed by Desnoyers show that manufacturers' specifications are extremely conservative since the endurance of many test blocks was two orders of magnitude greater than the specified rating. Pages within blocks perform at different levels and in order to stress the complete block, our experiments programmed all pages within each block and assessed the combined error count for the entire block.

# 4 EXPERIMENTAL DESIGN

The data required for the GP portion of this research was accumulated through the destructive testing of four NAND Flash memory devices. Blocks in each Flash device were initially p/e cycled over a period of one week in an oven which simulated real world usage for the devices. Following this simulation of usage, a 12 hour retention period was observed. This temperature accelerated phase of the process simulates 3 months real world retention. Data was gathered by performing error counts immediately before and after the retention period. Following this series of tests, the data acquired was used by GP to evolve our Retention Period Classifier. The procedures mentioned above will now be discussed in more detail.

## 4.1 Hardware

A purpose built hardware platform was constructed to facilitate the issuing of commands from a PC to perform detailed tests on Flash chips. A software program was then developed to use this hardware to perform detailed endurance cycling and retention period evaluations on NAND Flash memory devices. All details of the tests and data acquired from the tests were saved to a database for future reference.

## 4.2 Testing

Due to the time consuming nature of these tests and the number of available Flash testing units, four chips of the same model from the same manufacturer were tested. One requirement of the week long simulation of usage test is to allow blocks some recovery time between programs. This is because the speed with which a device degrades is directly related to the speed with which it is being cycled; a drive that is completely rewritten ten times in a single day will accumulate considerably more damage than one that is completely rewritten ten times in one week.

Table 1: Blocks were cycled to a number of different levels in order to gain a wide range of test data. Intervals of 5,000 cycles were used in the range of 5,000 to 30,000 cycles.

| Target Cycles | Test Blocks |
|---|---|
| 5,000 Cycles | 11 Blocks |
| 10,000 Cycles | 11 Blocks |
| 15,000 Cycles | 11 Blocks |
| 20,000 Cycles | 11 Blocks |
| 25,000 Cycles | 11 Blocks |
| 30,000 Cycles | 11 Blocks |

Taking this into account, as well as our aim of acquiring data at varying levels of cycling across many chips, 66 blocks were tested per device. These blocks were randomly chosen and divided into six cycling groups as shown in Table 1. The test software was written to balance the cycling rate throughout the week so that all cycling groups would run for the full duration of the process regardless of their target number of cycles. This meant, for example, that blocks with a target of 5,000 p/e cycles would have more recovery time between cycles than blocks completing 30,000 cycles.

Error counts were performed by writing a predefined pattern (a hexadecimal string) to a block and then reading it back, counting the number of errors introduced by storing the data in the block. During the temperature accelerated parts of the process, the blocks were cycled by iterating through a group of six pre-determined random hexadecimal string patterns. The pattern used to perform error counts at the end

Table 2: The testing procedure produced three inputs and one output for use with GP. Blocks were cycled for one week at temperature before the retention period was evaluated.

| Step | Duration | Temperature | Operation(s) | Result |
|---|---|---|---|---|
| 1 | 1 Week | 85°C | Repeated cycling of 66 blocks using a number of pre-defined random patterns. 1 cycle was performed on each block before moving on to the next block in the sequence. This allows some recovery time between cycles on each individual block. | Number of cycles completed is GP input 1. |
| 2 | — | 85°C | At the end of step 1, an error count was performed on all blocks using the difficult pattern. | Number of errors is GP input 2. |
| 3 | — | 25°C | Prior to the start of the 12 hour retention period, the chip was cooled to room temperature, with an error count performed on all blocks using the random pattern. | Number of errors is GP input 3. |
| 4 | 12 Hours | 85°C | The chip remained idle for the duration of the retention period. | — |
| 5 | — | 25°C | Following the 12 hour idle time, the chip was cooled to room temperature, and an error count was performed on all blocks using the random pattern. | Number of errors is GP output. |

of the week long cycling phase was supplied to us by our industrial partners. This pattern is referred to as the difficult pattern[3], and stressed blocks far more than regular data patterns due to the arrangement of bits (and in turn the charge stored in adjacent cells) required to represent the pattern. The pattern used to determine the number of errors in each block immediately before and after the retention period was a pre-determined random pattern.

The steps required during the overall process are listed in Table 2. All four test devices were placed in an oven and, when the oven temperature had stabilised at 85°C, the week long temperature accelerated cycling was initiated. During this period, data such as error counts and read, program, and erase times were recorded every 1,000 cycles for future reference. At the end of the week-long test, the difficult pattern was written to each test block and the corresponding error count was recorded.

The temperature accelerated retention period required that the blocks be programmed and read at room temperature. Following the completion of the week long test, the oven was cooled to 25°C and the pre-retention random pattern error count was performed. The oven was then reheated to 85°C and the chips were left idle at this temperature for 12 hours. Upon completion of this, the data was then read from all test blocks, and error counts performed, when the chips had cooled to 25°C once more.

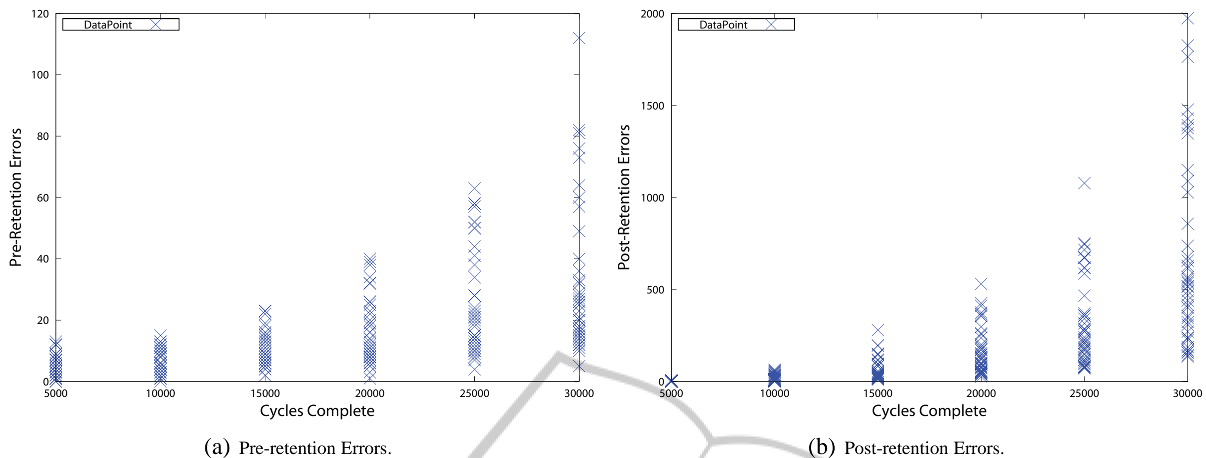
The four chips successfully completed this series of procedures, each providing data from 66 blocks for use in our GP experiments. The fact that it took eight days and the destruction of blocks on a number of Flash memory devices to accumulate just 264 data points stresses just how expensive this form of testing is in terms of both time and hardware.

Table 3: The input parameters for the GP system are the number of cycles performed and two error counts, each using a different pattern. A single output is produced.

| Parameter | Description |
|---|---|
| Input 1 | Cycles Completed |
| Input 2 | Pre-Retention 'Difficult Pattern' Error Count |
| Input 3 | Pre-Retention 'Random Pattern' Error Count[4] |
| Output | Post-Retention 'Random Pattern' Error Count |

[3]We have not included the hexadecimal string used in the difficult pattern due to a Non Disclosure Agreement with our partners in industry.

[4]Input 3 was later removed from the dataset as all values, regardless of cycles completed were 0.




(a) Pre-retention Errors.

(b) Post-retention Errors.

Figure 1: The error counts before and after the retention period. The different scales are to be expected as the number of post-retention errors should far exceed the number of pre-retention errors. The number of cycles has a significant effect on both the number of pre-retention and post-retention errors.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

## 4.3 Test Data

In order to evolve our Retention Period Classifier, we determined that the inputs and output for the GP system should be those shown in Table 3, since these are the quantifiable characteristics of the test blocks before and after the retention period. The first input is the number of cycles completed by the block at the end of the seven day temperature accelerated simulation of usage. The other two inputs are both error counts performed on the test block.

As expected (based on trial runs and related research), the data acquired before and after the retention period was extremely noisy, meaning experimental data points containing similar, or even identical inputs produced varied outputs. Furthermore, the pre-retention random error pattern resulted in zero errors on every test so this column was removed from our data set, leaving us with the other two inputs and one output already described.

Figure 1 shows the data acquired from the earlier tests. This shows the levels of variation in the data acquired from the temperature accelerated retention periods. For example, the pre-retention error rates at 30,000 cycles vary in the range of 5 to 112 errors, while the post-retention error rates for blocks cycled to 30,000 cycles vary from 136 to 1974 errors.

The Flash devices under test have a specified endurance of 5,000 cycles and a specified retention of 10 years. As mentioned previously, a trade-off exists between these two characteristics which is why devices cycled to higher levels than 5,000 (and up to 30,000) can reasonably be expected to successfully retain data over the relatively short time period of three months.

## 4.4 Decision Boundary for Binary Classifier

In a Flash memory device, a block is marked as bad (and removed from service) when the number of single bit errors is greater than the number of errors correctable by the on chip ECC. This is calculated as 11,915[5] correctable single bit errors for the MLC NAND Flash device described in Section 2.1.

For the purposes of this paper we have chosen 200 post-retention errors as the point at which a block is deemed to have failed. This number was chosen as our tests did not cycle any blocks to the level required to generate error counts of the magnitude of the maximum correctable bit rate. A decision boundary of 11,915 would lead to all blocks being classified as successfully retaining their data beyond the retention period and would not provide any useful data for training our GP system.

Using a 200 bit error point of failure results in all but one data point successfully passing the retention period up to and including 15,000 cycles. Most data points pass at 20,000 cycles, while most fail at 30,000. The 25,000 cycle data points are generally the most difficult to classify as approximately half the points fail and do not retain their data. These pass / fail rates at a decision boundary of 200 errors provide a good range of data to trial the evolution of our Retention Period Classifier. Table 4 shows the pass and fail rates for blocks when the decision boundary for correctable bit errors is set at 200.

[5] 4096 bytes per page * 128 pages = 524288 bytes per block. 12 bit correction per 528 bytes = 11915 correctable bits per block.

Table 4: The number of blocks which pass and fail at each level of cycling when the maximum number of correctable bit errors is set at 200.

| Cycles | Pass Blocks | Fail Blocks |
|---|---|---|
| 5,000 | 44/44 (100 %) | 0/44 (0 %) |
| 10,000 | 44/44 (100 %) | 0/44 (0 %) |
| 15,000 | 43/44 (97.73 %) | 1/44 (2.27 %) |
| 20,000 | 32/44 (72.73 %) | 12/44 (27.27 %) |
| 25,000 | 18/44 (40.91 %) | 26/44 (59.09 %) |
| 30,000 | 7/44 (15.91 %) | 37/44 (84.09 %) |
| Total | 188/264 (71.21 %) | 76/264 (28.79 %) |

## 4.5 Data Analysis

Analysis of the data set found that correctly classifying every point as being above (fail) or below (pass) the predefined decision boundary was impossible. This is due to the extremely noisy nature of the data with identical inputs from multiple data points resulting in differing outputs.

Data points which had identical inputs but outputs which led to different results were labelled *conflicting data points*. At 15,000 cycles, only one data point was regarded as a fail (a post-retention error count above the decision boundary of 200). However, this point was a conflicting data point, as another point with identical inputs resulted in a pass. This meant that 100% success when classifying data points at 15,000 cycles was impossible due to the existence of two conflicting data points as only one of them could be classified correctly.

Details of conflicting data points are shown in Table 5. The first data row in this table states that two data points contained identical inputs (15,000 cycles, 16 pre-retention errors) with one of the data points resulting in a pass while the other data point failed. The scale of this problem increased for greater levels of cycling with a total of 18 conflicting points across six sets of inputs (in some cases up to five data points had identical inputs) at 20,000 cycles. This resulted in at least seven data points which would be impossible to classify correctly. A minimum of six points would be impossible to classify correctly at 25,000 cycles while at least five data points would always be incorrect at 30,000 cycles due to conflicting results.

Further analysis examined data points that showed high potential for misclassification as they produced a different result to their surrounding neighbours. *Potential misclassifications* were data points which produced different results to other data points which contained similar inputs. When the majority of data points containing the same number of cycles and within a range of 10 pre-retention errors all resulted in the opposite classification to the point under review, the point was regarded as a potential misclassification.

Table 5: The number of conflicting data points at each level of cycling. Conflicting data points make 100% correct classification impossible as identical inputs produced different results. For example, the third data row in this table states that 5 data points contained identical inputs (20,000 cycles, 8 pre-retention errors) with four of the data points resulting in a pass while the other data point failed.

| | Inputs | | Result | |
|---|---|---|---|---|
| Count | Cycles | Errors | Pass | Fail |
| 2 | 15,000 Cycles | 16 | 1 | 1 |
| 2 | 20,000 Cycles | 5 | 1 | 1 |
| 5 | 20,000 Cycles | 8 | 4 | 1 |
| 2 | 20,000 Cycles | 10 | 1 | 1 |
| 5 | 20,000 Cycles | 11 | 2 | 3 |
| 2 | 20,000 Cycles | 14 | 1 | 1 |
| 2 | 20,000 Cycles | 26 | 1 | 1 |
| 2 | 25,000 Cycles | 8 | 1 | 1 |
| 2 | 25,000 Cycles | 11 | 1 | 1 |
| 4 | 25,000 Cycles | 14 | 3 | 1 |
| 5 | 25,000 Cycles | 15 | 2 | 3 |
| 3 | 25,000 Cycles | 21 | 1 | 2 |
| 2 | 30,000 Cycles | 12 | 1 | 1 |
| 2 | 30,000 Cycles | 14 | 1 | 1 |
| 2 | 30,000 Cycles | 15 | 1 | 1 |
| 3 | 30,000 Cycles | 16 | 2 | 1 |
| 2 | 30,000 Cycles | 25 | 1 | 1 |

At 15,000 cycles, one data point was prominent as a potential misclassification while at both 20,000 and 25,000 cycles this number increased to 12. The number of potential misclassifications then decreased to six at 30,000 cycles.

## 4.6 Genetic Programming

Rather than developing our own GP system, ECJ (Luke, 2010), a Java based Evolutionary Computation system was employed. Prior to beginning our GP runs, the data was randomly divided into 8 groups

in preparation for (*k*-fold) cross validation. We did, however, ensure that all cycling counts were evenly represented in all data sets.

A small number of preliminary test runs were completed in order to experiment with different GP parameter settings and function sets. These runs examined the results for varying numbers of generations, population sizes, tree depths, and reproduction, crossover and mutation rates. A number of combinations of various operators in the function set were also tested. Following these trial runs, the GP parameters listed in Table 6 were chosen. The GP function set comprised seven functions, while the terminal set comprised the two inputs discussed earlier and also five constants.

Table 6: Tableau showing GP parameters and settings.

| Parameter | Details |
|---|---|
| Objective | Correctly classify data points as pass or fail using a decision boundary of 200. |
| Terminal Set | x, y, 2, 3, 5, 7, 11 |
| Function Set | +, -, *, /, sin, cos, tan |
| Fitness | The percentage of data points correctly classified. |
| Hits | The number of data points correctly classified. |
| Generations | 100 |
| Population | 1000 |
| Max Tree Depth | 15 |
| Crossover Rate | 0.8 |
| Mutation Rate | 0.15 |
| Reproduction Rate | 0.05 |

The GP system operated as a binary classifier with the fitness of each individual deemed to be the percentage of data points classified correctly. The number of correct classifications for each individual were also recorded and are referred to as hits. Each data point was classified by first calculating the projected output using the evolved individual. If the individual resulted in output greater than the decision boundary of 200, the individual predicted a fail, while an output of less than the decision boundary predicted a pass. This was then compared to the actual output for the data point to determine if the evolved individual's classification was correct.

In order to verify whether the proposed GP process would generalise well across the cycling levels tested, a variant of *k*-fold cross validation was performed. The standard *k*-fold cross validation technique requires that the available data is evenly divided into *k* groups. Following this, *k* GP runs are performed with a different group being used as validation data for each run while the remaining *k*-1 groups are used as training data. The leads to the GP system training on *k*-1 groups with the best individual from each run being validated using the previously unseen data group set aside for validation. Upon completion of the *k* runs, the validation results from the *best-of-run* individuals are averaged to give a representation of how well the system can generalise to solve the specified problem.

# 5 RESULTS

As mentioned in the previous section, the 264 data points were divided into eight groups. This resulted in each group containing the same number of points. A variation of the standard *k*-fold cross validation was then used by splitting the training groups into two parts. Having assigned one of the eight data groups for validation in each data set, the remaining seven groups were randomly divided into five training groups and two testing groups. This resulted in each data set using 62.5% of the data for training, 12.5% for validating every individual from the final population on unseen data, and 25% for testing of the best-of-run individuals. The extra validating step assessed the ability of the entire population at the end of the run to generalise to previously unseen data points since there is no guarantee that the individual which performed best on the training data will also generalise the best.

The best-of-run individual following the validation stage of the process was chosen to go forward to the testing stage. In the case of a number of individuals achieving the same best-of-run result, the shortest individual was chosen since it is expected that shorter individuals will generalise better. Since 8-fold cross validation requires a minimum of just 8 runs – 1 per data set – 10 runs were performed per data set giving a total of 80 runs.

Table 7 shows the results from the 8-fold cross validation runs. An average of 79.09% data points were classified correctly by the best-of-run individuals on the previously unseen testing data across all 80 runs, while an average of 85.79% data points were classified correctly by the best performing individuals from each of the eight folds. Despite the difficulties posed

Table 7: Results from 8-fold Cross Validation test. Data was divided into 62.5% training, 12.5% validation, 25% testing data. Fitness refers to the percentage of data points classified correctly while hits refers to the number of correct classifications. The hits column also shows the total number of data points for each subset of the data. The last row in the table shows the average number of data points classified correctly by the best-of-run individuals.

| | Training (62.5%) | | Validation (12.5%) | | Testing (25%) | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean Pop | | Mean Pop | | Mean | | Best | |
| Fold | Fitness | Hits | Fitness | Hits | Fitness | Hits | Fitness | Hits |
| 0 | 87.24% | 143.95/165 | 78.46% | 25.89/33 | 78.03% | 51.50/66 | 86.36% | 57/66 |
| 1 | 89.30% | 147.34/165 | 80.36% | 26.52/33 | 76.36% | 50.40/66 | 80.30% | 53/66 |
| 2 | 86.30% | 142.39/165 | 87.14% | 28.76/33 | 76.97% | 50.80/66 | 84.85% | 56/66 |
| 3 | 89.76% | 148.10/165 | 72.93% | 24.07/33 | 76.82% | 50.70/66 | 87.88% | 58/66 |
| 4 | 87.79% | 144.85/165 | 77.30% | 25.51/33 | 83.79% | 55.30/66 | 89.39% | 59/66 |
| 5 | 87.25% | 143.97/165 | 89.58% | 29.56/33 | 80.30% | 53.00/66 | 84.85% | 56/66 |
| 6 | 87.86% | 144.96/165 | 82.13% | 27.10/33 | 83.33% | 55.00/66 | 89.39% | 59/66 |
| 7 | 87.67% | 144.66/165 | 70.42% | 23.24/33 | 77.12% | 50.90/66 | 83.33% | 55/66 |
| | 8-fold Cross Validation Average: | | | | 79.09% | 52.20/66 | 85.79% | 56.63/66 |

by the data set (as discussed in Section 4.5), the overall best performing individual achieved 89.39% correct classification on previously unseen data.

Considering the extremely noisy nature of the data points (potential misclassifications) and the fact that some identical inputs produce differing classifications (conflicting data points), these results show that the GP process can be used to generate a Retention Period Classifier which will correctly classify the majority of test cases.

Since no other approaches to predict the retention period for Flash memory blocks have been reported in the literature, comparative evaluation with previous results was not possible. Section 6 includes details of plans to employ other machine learning techniques to provide benchmarks against which the GP system proposed in this paper can be compared.

The levels of correct classification achieved in this first set of experiments show the huge potential of this prediction technique. Future research will aim to further validate and improve the initial results reported here in order to develop a classifier suitable for real world application.

## 6 FUTURE WORK

Taking into account what we have learned from this research, we propose to perform another batch of experiments but to focus on higher levels of cycling. The maximum number of post-retention errors found during the data acquisition phase of this experiment was 1,974. In our next test, we will focus on intervals of 5,000 cycles starting at 10,000 and proceeding up to 50,000. Judging by our test data and the results obtained, we believe that this will allow us to evolve an improved Retention Period Classifier by allowing GP to train the population using a real world number of correctable bit errors.

As part of the next iteration of this research we will also generate classifiers using a number of other methods such as neural networks and support vector machines. The results obtained will provide data allowing a comparative evaluation between a number of different machine learning techniques.

This is just one task of a larger research project. The overall goal of our research is to improve the performance of SSDs using EA techniques. We aim to achieve this by improving the endurance of the NAND Flash memory used by the vast majority of these drives. A Genetic Algorithm (GA) will be used to evolve optimised parameter settings for Flash memory devices under test. As mentioned earlier, there is a trade-off between endurance and retention. The classifier evolved in this paper will be used in forthcoming research to predict whether blocks will pass a temperature accelerated retention period when controlled by modified parameter settings. This will potentially save a lot of time and avoid performing unnecessary tests for parameter settings which will fail.

## 7 CONCLUSIONS

This research set out to use GP to evolve a function to aid the prediction of blocks that would not retain data for a particular length of time. This has huge potential for use in SSDs as it aids the early identification of blocks that are likely to lose their contents when powered off. Initial analysis of the data acquired through the destructive testing of blocks within a Flash device showed the extremely noisy nature of the data including conflicting data points and the huge potential for misclassification, making it impossible to correctly classify all data points.

A form of 8-fold cross validation was used to verify that our GP process could evolve individuals capable of generalising to unseen data. A data division of 62.5% training, 12.5% validation, and 25% testing was used and 80 runs were performed to generate a variety of potential solutions. The best individual from our set of potential solutions achieved 89.39% correct classification on previously unseen data while the average result was 85.79%. This highlights the huge promise shown by this technique considering the difficulty posed by the noisy data set.

Since no method to predict the deterioration of Flash memory blocks over the course of a retention period currently exists, we believe this classifier has potential for real world application. We will continue to expand on the research introduced in this paper and build a more robust Retention Period Classifier by accumulating more data points at higher levels of cycling for use by GP.

We have confirmed that it is possible to use EAs to model Flash memory characteristics and now intend to explore this area in greater detail. In future research, we will progress to using EAs to improve the endurance of Flash devices by optimising the operating parameters for MLC NAND Flash memory.

## ACKNOWLEDGEMENTS

The authors would like to thank Barry Fitzgerald for his assistance in performing the temperature accelerated tests required for this research.

## REFERENCES

Aritome, S., Kirisawa, R., Endoh, T., Nakayama, R., Shirota, R., Sakui, K., Ohuchi, K., and Masuoka, F. (1990). Extended data retention characteristics after more than 10,000 write and erase cycles in EEPROMs. In *International Reliability Physics Symposium. 28th Annual Proceedings.*, pages 259 –264.

Aritome, S., Shirota, R., Hemink, G., Endoh, T., and Masuoka, F. (1993). Reliability issues of Flash memory cells. *Proceedings of the IEEE*, 81(5):776 –788.

Brewer, J. and Gill, M. (2008). *Nonvolatile Memory Technologies with Emphasis on Flash (A Comprehensive Guide to Understanding and Using Flash Memory Devices)*. Wiley-IEEE Press.

Chen, F., Koufaty, D. A., and Zhang, X. (2009). Understanding intrinsic characteristics and system implications of Flash memory based solid state drives. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS ’09, pages 181–192. ACM.

Desnoyers, P. (2010). Empirical evaluation of NAND Flash memory performance. *SIGOPS Oper. Syst. Rev.*, 44:50–54.

Grupp, L. M., Caulfield, A. M., Coburn, J., Swanson, S., Yaakobi, E., Siegel, P. H., and Wolf, J. K. (2009). Characterizing Flash memory: Anomalies, observations, and applications. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 24–33, New York, NY, USA. ACM.

Kahng, D. and Sze, S. (1967). A floating-gate and its application to memory devices. *The Bell System Technical Journal*, 46(6):1288–1295.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT press.

Luke, S. (2010). ECJ 20. A Java-based evolutionary computation research system. http://cs.gmu.edu/∼eclab/projects/ecj/.

Mielke, N., Belgal, H., Fazio, A., Meng, Q., and Righos, N. (2006). Recovery effects in the distributed cycling of Flash memories. In *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, pages 29 –35.

Mielke, N., Marquart, T., Wu, N., Kessenich, J., Belgal, H., Schares, E., Trivedi, F., Goodness, E., and Nevill, L. (2008). Bit error rate in NAND Flash memories. In *Reliability Physics Symposium, 2008. IRPS 2008. IEEE International*, pages 9–19.

Pavan, P., Bez, R., Olivo, P., and Zanoni, E. (1997). Flash memory cells - an overview. *Proceedings of the IEEE*, 85(8):1248 –1271.

Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu.

Shread, P. (2009). Fusion-io lowers the price of solid state storage. http://www.enterprisestorageforum.com/hardware/news/article.php/3829246/Fusion-io-Lowers-the-Price -of-Solid-State-Storage.htm.

Sullivan, J. and Ryan, C. (2011). A destructive evolutionary algorithm process. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15:95–102. 10.1007/s00500-009-0513-2.

Yaakobi, E., Ma, J., Grupp, L., Siegel, P., Swanson, S., and Wolf, J. (2010). Error characterization and coding schemes for Flash memories. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1856 –1860.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS