A Destructive Evolutionary Process A Pilot Implementation

Joe Sullivan
Department of Electronic and Electrical Engineering
Limerick Institute of Technology
Moylish, Limerick
+353 61 208339
Joe.Sullivan@lit.ie

Conor Ryan
Department of Computer Science & Information Systems
University of Limerick
Castletroy, Limerick
+353 61 202755
Conor.Ryan@ul.ie

Abstract

This paper describes the application of evolutionary search to the problem of Flash memory wear-out. The operating parameters of Flash memory are notoriously difficult to determine, as the optimal values vary from batch to batch. These parameters are usually established by an expensive, once off process of manual destructive testing at design time. Testing on individual batches is normally not feasible. We establish the viability of a platform that performs destructive experimentation on hard silicon, using a Genetic Algorithm to automatically discover optimal operating parameter settings. The results demonstrate a minimum average life extension of between 250% and 350% over the factory set read write and erase conditions with a maximum life extension exhibited of 700% for cells within the same device. It was necessary to build specialized hardware to perform the repetitive testing required by the GA, here we describe this hardware and demonstrate how the lessons learned in this pilot study will allow us to proceed with a more complex parallel evaluation platform, which will facilitate a larger problem space, larger population size and diversity of search techniques, facilitating the near no cost life extension of a split-gate Flash memory device.

Categories and Subject Descriptors

I. Computing Methodologies---I.2. Artificial Intelligence--I.2.8 Problem Solving, Control Methods, and Search-**Subject descriptor:** *Heuristic methods*

B.Hardware--B.3—Memory Styles--B.3.1 Semiconductor Memories -**Subject descriptor:** *Read-only memory (ROM)*

General Terms

Experimentation.

Keywords

Silicon Design, Reliability, Experimentation, Flash Memory.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
Copyright 2007 ACM 978-1-59593-697-4/07/0007...$5.00.

1. Introduction

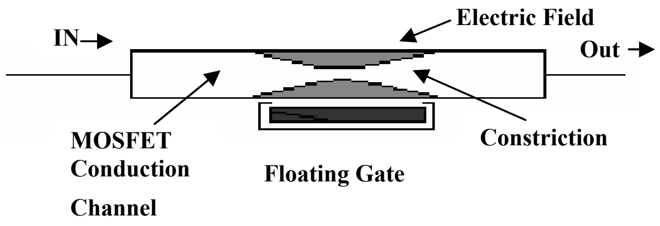
The primary function of a memory device is to retain settings. The purpose of EEPROM or E^{2}PROM (Electrically Erasable Programmable Read Only Memory) is to retain settings after the power has been removed and to be electrically alterable in circuit when the power is applied. In recent years, floating gate memory devices have become the overwhelming technology of choice for those applications that require non-volatile semiconductor memory [1,9,11]. There are three broad categories of EEPROM [1,11,17]. Arrays in which each byte of memory is erased independently (byte erasable), arrays in which all elements are erased together (bulk erasable), and Flash memory. Byte erasables have erase circuitry associated with every byte. This make them unsuitable for large arrays as the extra complexity required takes up valuable silicon real estate. Bulk erasables are unsuited to many applications due to the requirement to erase all location in order to alter the data in a single memory byte. Flash memory, so called because segments or blocks of memory are erased or flashed together, is the ideal compromise of space and usability. It is Flash memory that is the focus of this paper.

Flash memory is found in almost every consumer electronics product from mobile phones, where it is used to store text messages and phone book information to TVs, Car Radios, fridge freezers and so on. Anything in fact that needs to remember a setting between power down and power ups. Flash has been used to great effect in recent years in memory sticks, MP3 players and game consoles and has huge potential in the nascent, PC semiconductor permanent storage market, where it will in the future replace mechanical disks as the main permanent storage element. The great strength of semiconductor memory is that is has no moving parts and very low power consumption. This make Flash very attractive for mobile application where its great resilience and compactness are huge advantages.

Flash memory is currently undergoing rapid development [1,11] and a number of differing approaches and cell architecture are currently in use [14], however all Flash memory utilises floating gate technology using some specialised means to get electrons on and off the floating gate. Specifically, this research uses Split Gate Field Enhancing Tunnelling Injector EEPROM cells [2,11]. For more detail see IEEE document 1005-1998. This cell is particularly suited to this evaluation due to its reduced susceptibility to being 'over erased', where excessive erasure inhibits reprogramming of the cell and the absence of an 'erase disturb' mechanism, where programming one cell in a memory array, results in the programming or erasure of another nearby. Furthermore program disturb is not exacerbated by accumulated erase and programming cycling [2,11].

The floating gate memory element consists of an electrode that is completely encased in an insulator, which acts as the gate electrode to a MOSFET (Metal-Oxide Semi-conductor Field Effect) transistor. The electric charge on the floating gate constricts the flow of electrons in the conduction channel, which is the memory sense element. Conduction in the sense element above and below a predefined threshold current yields logic One or Zero. Although the floating gate is entirely surrounded by an insulating material, by applying atypical voltage conditions it is possible to induce *Hot Electron Injection* [3,18] and *Fowler-Nordheim* tunneling [6] to control the charge on the floating gate and thus alter the state of the memory element. In essence we use high voltages to accelerate and punch electron through the very thin insulator to reside on the floating gate where they influence the conduction of the MOSFET transistor.

Figure 1. Idealised Schematic of MOSFET



A state machine or microprocessor is employed to apply the write and erase special voltage conditions that will alter the state of the memory element. Neither the write nor the erase cycles are benign processes. Both exact a toll on the devices ability to be re-programmed by the degradation of the silicon through electron trapping in the oxide. As the electrons are punched through the insulating oxide some electrons get trapped, obstructing the path of subsequent electrons [9,14]. The accumulated trapped charge in the oxide eventually makes the cell un-programmable, that is, the cell's state can no longer be altered. The specified endurance value of the memory matrix is a measure of how many write and erase cycles the device can sustain, before it becomes un-programmable. Typically this is between 10k and 100k [14]. This is a critical metric of the device. If wear-out can be minimised then the life of the part may be more favorable specified.

There are parameters which effect the wear-out rate of the memory device such as erase duration, rate of change of potential, programming current and so on [10]. These parameters are often interdependent and are ascertained by exhaustive testing during design, testing and qualification of the part. Once optimum values have been established they are embedded in to each device for use during the lifetime of the memory chip. This process is a one-time effort and covers all manufactured device of this part number. It is a very expensive undertaking in terms of manpower and equipment, but takes no account of manufacturing variations or the condition of the device as it progresses through its life in the field. This is reflected in specification sheet endurance values which are degraded in order to guarantee the worst-case scenario.

Variations in silicon occur for many reasons. Manufacturing of the devices may take place in many plants around the world. Furthermore each batch of parts may be processed under slightly differing conditions within the same manufacturing plant. Variations also occur from wafer[1] to wafer within a batch. This is a hugely expensive exercise, in both time and money, so the factory-calculated values remain static under all circumstances [11]. It is however, unlikely that this "one hat fits all approach" will be the optimum set of values for each device or subset of devices. This paper is concerned with developing a GA driven approach that is cheap and fast enough to be applied to subsets of devices.

2. Method

We set out to prove that it is a viable prospect to use evolutionary search to conduct read, write and erase experimentation on Flash memory silicon, to ascertain if there are optimum values for parameters, that if coded into each device in the batch, will result in a life extension for those devices. A GA was chosen for the evaluation because of its ability to succeed in noisy environments [4], and where the search space may not be well understood [5,17]. There are a number of reasons why this search problem is not suited to special-purpose search techniques such as hill climbing and simulated annealing. Firstly the interdependence of the variables at play for each specific device and batches of devices is unknown. Secondly the absolute endurance of each device is variable. This effectively creates a noisy environment in which there is no assurance that the space is continuous or linear. Finally the evaluation will have to be performed repeatedly. That is for each group or batch of devices such as 'all devices on a wafer', or 'all devices in a specific production run' and so on.

An offer of access to a Teradyne test platform (a generic industrial electronic component tester used in the manufacture of silicon chips) from the supplier of the chips we tested was turned down due to fears of vagaries in production pressure adversely affecting the project schedule, instead a hardware platform was conceived that would give the project maximum flexibility for change as the information was revealed and that would remove as much software tasks from the silicon under test as possible. This would allow us retain as much of the Flash memory for the experiments as possible. A GA was utilized to drive the search and generate the experimentation, which were conducted on the hardware platform in real time.

The platform conducts read write and erase cycles who's state machine parameters have been coded to the individuals in a population of GA generated solutions. The fitness of each solution is established by the indirect detection of the memory cells threshold voltage or by the cell's absolute longevity. This fitness information is used to generate the next generation of experiments in the normal way.

High Level Software Algorithm.

1. **GA creates an initial population**
2. **Sends entire population through the serial port to the control board**
3. **Control board schedules each individual to the DUT (Device Under Test)**

[1] Wafer, An eight to twelve inch circular slab of silicon containing hundreds of silicon devices. These are tested by probing. The slab is then broken up and the working chips are fitted to housings. The non-workers are discarded.

4. **DUT performs write and erase cycling to the memory location indicated, under the conditions set by the control board**
5. **DUT indicates that it has failed to erase**
6. **Control board records this results and calculates the fitness**
7. **Control board returns fitness figures to the PC**
8. **PC writes report files and returns next generation to the control board**

Figure 2. Block Diagram of the Pilot Hardware Platform

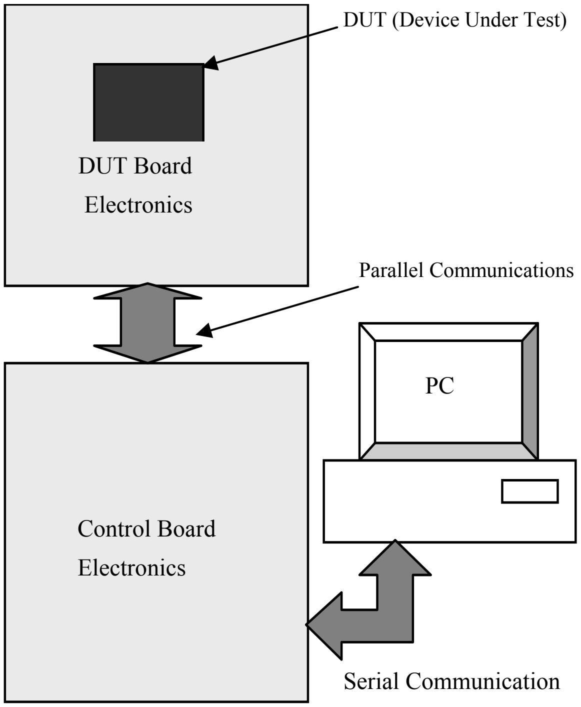
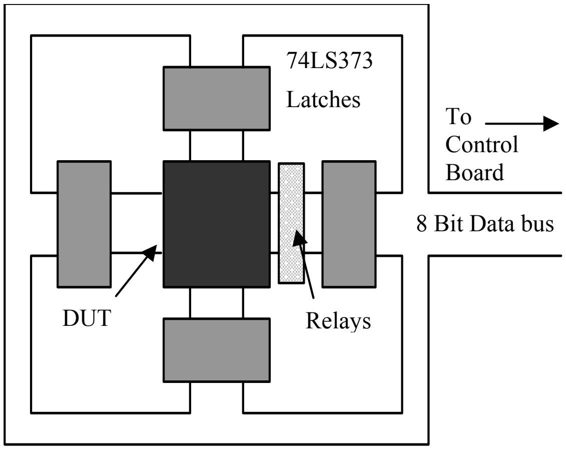


Figure 2 shows a high level overview of how the GA communicates with the silicon memory. The GA creates a population of strings, which are coded to control registers within the silicon. These registers are responsible for setting the precise read, write and erase conditions that will apply during programming and erase cycles. There are Five 'Eight bit' registers which control such things as Erase Time (the length of time the erase condition will apply), High voltage set up time (rate of change of applied voltage), Erase Current and so on. In a more complex arrangement these conditions may be imported directly to the memory elements from external precision voltage and current sources. The control board electronics shown above has a large amount of responsibility and contains an 8051 microprocessor as well as a number of A to D (Analog to Digital) converters D to A converters, a watchdog timer, extra memory and IO ports of mention just some of its functionality. We use this board to conducts communications with the PC via a serial port. It stores the population of experiments, cycles the DUT (Device Under Test) through each one, and stores the resultant fitness values. When the population has completed, the control board will return the fitness array to the PC where a new generation of experiments will be created. The DUT has a short monitor program, which is blown to the DUT before commencement of experimentation. The control board is also responsible for loading this code set to the DUT prior to starting experimentation. Using the control board heavily allows us to lighten the load on the DUT and so shorten this code set preserving maximum flash memory space for experimentation.

The DUT board shown in figure 3 consists of number of eight bit latch devices; each latch is assigned to a subset of pins on the memory device. The control board can write any value to theses latches and they will maintain this binary value on the pins of the DUT. Thus the control board can systematically write any combination of 0 and 5 volts to the pins of the DUT in order to communicate with it, control its operation and its operating mode.

The DUT board has a mode of operation in which the memory cell current (Cell I) is outputted directly through the pins on the DUT. This port requires isolation to facilitate the careful measurement of the cell's threshold current without interference from other current source or sink devices. This is achieved through the use of high quality relays connected directly to the port pins. Again the control board can put the hardware into this mode of operation and also "Reset" the device out and back into normal memory mode.

Figure 3. Block Diagram of the DUT Board



The control board has a large number of discrete tasks to perform all under the control of a PC based high level program. A large number of other hurdles had to be overcome in order to make it the whole system work.

Namely,

1) Devise a compact embedded code set for the memory device that will run the low level chores including communications.

2) Devise a Programmer that will program the memory device with the embedded code set.

3) Devise an embedded code set for the control board that will communicate with the PC and the DUT. It will also transmit their

fitness values to the PC when complete. This firmware must also reset and control the modes of operation of the memory device.

4) Conceive a PC program that will integrate a GA and the communication protocol and record all results in a format fit for analysis.

An exhaustive treatment of the hardware and software including schematics, code listings and explanations is available for download on FTP://193.1.89.105

2.1 Experiment Design

The purpose of this segment of research is to establish the shape of a viable hardware and software platform. The pilot hardware is therefore necessarily limited in scope in a number of ways. This creates a challenge in devising experimentation within these limits that will yield as much good information as possible about the desirable features of any future hardware platform. A case in point is the organisation of the memory array. In Flash memory, groups of memory cells are erased together. This limits the number of discrete erase plans that can be tried in any one piece of silicon. In this memory device there is a 10 kilobyte code memory array grouped in blocks of 64 bytes and a 640 byte data memory array grouped into blocks of 4 bytes. We cannot use data and code memory sections within the same evaluation, which means that there are 160 discrete erasable blocks available to any one experiment. This will limit the population size and or the number of generation within any erase based analysis.

There are further limitations based around the available values for variables. For example, an onboard 8-bit register controls the erase time. As the binary value in the register increases so too does the erase time. Therefore the erase time is variable in 256 discrete steps only, and has specific minimum and maximum values. To access a fuller spectrum of values would require specialised high precision equipment to be incorporated from an external source which can bypass the internal arrangement by the direct control of the parameters, such as erase time, programming current and so on. This approach was rejected on the basis of complexity and cost although the ability of the built hardware to implement this method has been retained.

The search space is large given the cost of evaluation in terms of silicon and experiment time. It was felt that the appropriate response to such limitations was to proceed in a piecemeal way, restricting the search space if necessary and using small population methods [9], until a viable way forward emerged which would allow us to release the full power of the search technique. For example many of the parameters increment linearly with their control register value and we make the assumption that the values written to the control registers need never decrement over time and that the next value written to the registers should always be greater than the last. This derives from the assumption that it will never become less difficult to drive electrons on and off the floating gate since as time goes on, more oxide trap site occurs and injector degradation progresses. Secondly we restrict the starting point of the variables to any value below the halfway point to try to exclude random, narrowly focused solutions, likely to waste resources. This limiting approach is not ideal, in that we cannot assume that in the presence of other interdependent variables that the problem is linear. However this approach does yield the maximum amount of useful information and reduces experiment time and is therefore appropriate in the context of this pilot.

Each variable would also be explored independently to suggest good candidate variables to be left out of the first round of analysis. A literature review was conducted to help in this task. The results of the pilot would lead us to a hardware design that will allow us to un-restrict the search space later.

Within each device we would need to establish a base line against which to measure success. The most appropriate measure in this case is to use the factory-evaluated solution on a sample number of cells, and then perform the new evaluation on the rest of the cells within the same device. One problem with this is that it was possible that cells from one location on the silicon would performed better than others due to physical variation based on location. This required us to evaluate any endurance gradient that existed within the memory range. To establish the extent of this we sampled many cells throughout the address range and selected various locations to run the control group to see if any patterns developed. In the event no significant pattern was discernable.

A minor consideration is to establish if significant de-trapping [16] was occurring between cycling of the cells. De-trapping is where trapped charged particles are released form the oxide layer causing a partial resurgence in the device. This occurs in practice in the field and is taken into account somewhat in establishing base line endurances of parts, if not the spec sheet values themselves.

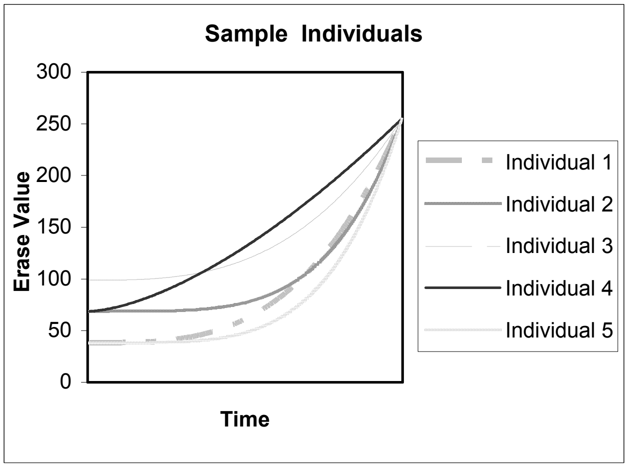
Finally, due to the black box nature of the DUT we needed a series of experiment that would verify the integrity of the platform and to build certainty that the derivatives were valid for the memory cells in play.

2.2 Experiments

Due to the adoption of a piecemeal approach discussed in 2.1 many of the experiments were modified or suggested by the results of previous ones. A preliminary round of experiments verified the integrity of the system and confirmed that the erase time function was dominating the degradation of the memory cells. This is in line with expectations arising from the literature review [1,6,9,11] and interviews with industry experts. The erase time also dominates the write/erase cycle time, the erase cycle being up to 31 Milliseconds long while the write cycles is between 30 and 122 Microseconds. From this it is likely that any erase time evaluation will dominate experiment duration also.

Erase time is clearly an important parameter and so the second round of experimentation detailed here focused on this. The GA was coded to the slope of the erase time and to the starting erase time point. Any erase plan that results in a failure to erase will terminate the solution with the fitness being derived from this termination point. The fitness is returned as an 8-bit integer. The multiples of write/erase cycles are scaled to ensure that we do not run out of erase steps before the natural end of the cell's life.

Figure 4. A sample of Individuals from generation One showing starting point and rate of change of the erase time



On each device we first process a range of cells with the factory default case in order to create a control group. Next we allowed the GA to choose a starting point and a rate of increase for the erase time. Figure 4 shows a random selection of a typical population of erase solutions. In each case above a starting erase time is shown at the left most side of the graph as well as the rate of increase of erase time up to the maximum of 256 (the longest erase of 31Milliseconds).

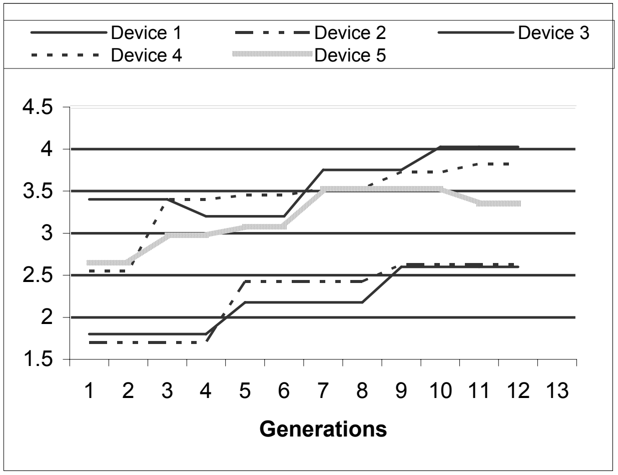
A single evaluation was confined to a single chip and so the total erase groups available to the experiments were 160. An initial population size of 20 would give us a maximum of 7 generations allowing 20 erase blocks for the control group. A uniform mutation rate was set at 5% using Roulette selection and single point crossover. The principle in operation is that any slope that is excessively weak during early life will result in an erase fail and a poor fitness value. Any slope that is excessively strong during early life will result in excessive wear and also result in a premature fail.

Five devices were tested to destruction in this way with variations in exact implementation in order to extract other useful information. For example several devices have two separate evaluations in order to intersperse the control group so as not to take the control group from any specific location in the device. Also device 3 was first cycled to destruction and re-run several months later in order to find out if significant de-trapping was occurring. In the next round of experiment this method of starting point and slope was expanded to include a number of other variable. The results are promising and are detailed in the next section

2.3 Results

In the following diagrams each data point represent the average of 20 individuals from each generation. The graph in Figure 5 shows the average absolute life of evolved solutions divided by the average control group values in each case. In other words the graph does not show the new extended endurances, it shows how much better the evolved solution is over the default factory solution applied to cells from the same device. In the final generations we see average life extensions for the evolved group of 250% to 350% over the control groups. This represents solutions that are at least 20 times better than the spec sheet value. Also apparent from the graph are increases in fitness over subsequent generation. A further suggestion from the data is that significant de-trapping has occurred in device 3 in the 5 months since it was first cycled to destruction.

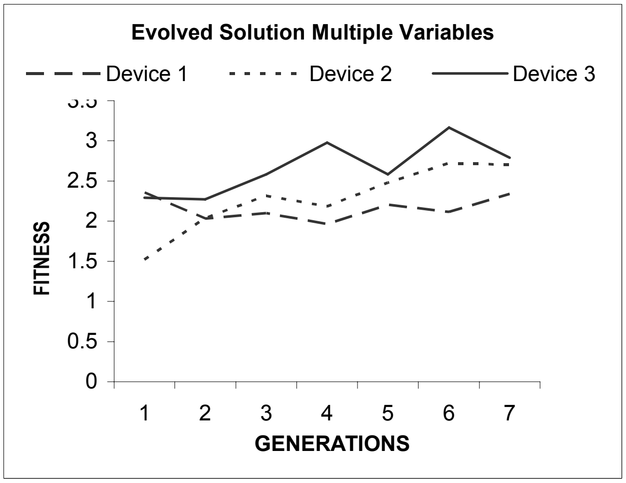
Figure 5. Showing Mean Endurance improvement over



Also indicated from the data is that a pre-degraded part (device 3) undergoes the same pro-rata increase in endurance as does a new part even though its absolute endurance is less than half that of a new cell. This suggests that the method will optimise parts originating from inferior processes.

The next round of experimentation extended the use of this start point and slope method to multiple variables applying the same proportionate starting point and slopes to the additional variables. The results are shown in Figure 6.

Figure 6. Mean Endurance improvement. Starting point and slope now applied to Multiple variables



Again we see a marked increase in endurance over the period of the experiment. However in this case the solutions are no better than the erase time only evolved solution. Having described the results what does all this uncover? The results point to a ground breaking increase in endurance figures. The best of our evolved memory cells are living 35 times longer than the specification sheet endurance value. The worst average figure in the final generations is 20 times better than the spec sheet value. This type of analysis is not carried out during probe or final test manufacturing due to the cost per device implications of the manpower and tester platform resources required to do this manually. Our approach is autonomous, so it does not require an expensive production tester and is automated so does not incur engineering cost. The current process of establishing programming parameters is crude and consequently there is excessive guard-banding of the spec sheet endurance value. This research shows us how we may engineer out the guard band by using refined computerised search.

Also evident from the data is that there are significant improvements that can be made to the hardware. Firstly, it is clear that the time taken to process the silicon is far too long. Some of this is accounted for by the fact that the memory chips endurance specification is, for factory operational reasons, excessively de-rated. However laboratory measurement showed that the experiment time is dominated by communications protocol latency. Lab measurements show that communications latency is eight times the time spent erasing. Checks on report files confirm this with reports spanning 28 hours showing only 3.5 hours spent erasing. This can be exploited to improve the performance of the platform. The problem is the convoluted manner in which the DUT relates to the G.A. The reason for that approach was to devolve as much responsibility to the control board as possible in order to reduce the complexity (and thus size) of the DUT embedded code. This preserves Flash memory for solution testing and indeed the embedded code resides in a single erase block of memory on the DUT (64 bytes). Having the DUT perform more tasks unilaterally will eliminate the latency. In so doing we must use more onboard memory but this trade off will pay off handsomely. There is a serial port on the DUT and so the control board can be eliminated entirely from the experimentation, retaining the control board only to program the DUT. At the same time the serial port baud rate can be greatly increased as the serial port in the DUT is more advanced that that of the control board.

Implementing this will greatly reduce the complexity of a single DUT site, as no control board is required. This opens up the use of parallel operation, the processing of multiple individuals at the same time I.E multi-site experiments. Another boon for this method is the expansion of the available erase groups that the GA can access. This will allow a greater number of individuals and generations and thus facilitate the expansion of the search space.

In Reeves [15] it is suggested that good results can be expected from a GA utilizing small populations especially if the starting conditions have defined characteristics. In Grefenstette [7] it is suggested that population sizes of less than 30 are quite adequate in many cases. The systematic selection of an initial population will allow us to improve the efficiency of the GA utilizing small population sizes of L=Yn where Y is between 1 and 2 and n is the string length [15]. This coupled to the expanded experiment space builds confidence that GA will be an effective search tool in this instance

Other positives that can be taken from this data is that no particular endurance gradient was detected within any single part, although this will need further verification. Furthermore the methods would seem to be valid for parts batches of differing quality. Pursuing this on a large scale may require the mass degradation of a portion of the available chip stock. There is an operating mode that will allow this to happen quickly although it may be necessary to augment the on chip charge pump which may not be able to facilitate mass erase cycling.

Improvements that can be easily made to the system can be summarised by two statements

1. The time taken to perform a complete evaluation can be greatly shortened
2. We can greatly increase the number of discrete experiment points available

These changes will allow us to proceed with a more complex parallel evaluation platform, which will facilitate a larger problem space, larger population size and diversity of search techniques, facilitating the near no cost life extension of a split-gate Flash memory device.

3. Conclusions

This paper has described the principle of operation of the newest and most promising form of non-volatile memory (flash). We have considered a key failure mechanism of Flash and illustrated a weakness in the current approach in dealing with this failure mechanism. We then endeavored to evaluate the prospect of applying a modern Evolutionary Search technique to the problem. We have successfully designed and build a machine that can evaluate silicon memory and conceive a programming plan tailored to that batch of chips. We have seen results with this system that show cells living 35 times longer than the specification sheet endurance value say they will live and have shown average results that are not less than 20 times better than the spec sheet. But most of all we have elicited information and requirements from the exploration of the problem, which provide a path forward to a fully implemented system. This pilot study has uncovered the critical hardware and software features that will facilitate the application of evolutionary search to Flash memory wear-out.

3.1 Further work

A multi-site prototype site has been built and a new embedded code set for the DUT has been written with fast serial communication connecting directly to the PC. Some evaluation and verification work has been done on this and a new 16-site version is under construction.

4. Acknowledgments

Our thanks to our industrial contributor particularly those in final test and design evaluation. Special thanks to the interviewee for their time and patients.

5. References

[1]Aritome, S. Shirota, R. Hemink, G. Endoh, T. and Masuoka, F. “Reliability issues of Flash memory cells,” *Proc. IEEE*, pp. 776–788, 1993.

[2]Bhattacharya,s.,et al., “Improved Performance and Reliability of split gate Source side Injected flash memory cells, IEDM tech digest pp339-342 1996

[3]Chung, S. Cherng-Ming Yih, Shui-Ming Cheng, Mong-Song Liang “A New Technique for Hot Carrier Reliability Evaluations of Flash Memory Cell After Long-Term

Program/Erase Cycles," IEEE Transactions On Electron Devices, Vol. 46, No. 9, September 1999.

[4]Fitzpatrick, J. Michael Jmf@Vuse.Vanderbilt.Edu) John J.Grefenstette + Grefenstette@Vuse. Vanderbilt.Edu) Genetic Algorithms in Noisy Environments *Computer Science Department, Vanderbilt University, Nashville, Tennessee 37235*

[5]Forrest, S. Mitchell,M. What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation Machine Learning 13, 285-319, 1993.

[6]Fowler, R.H., and Nordheim,L.,"Electron Emmission in Intense Electric Fields," Proceedings of the Royal Society of london, vol.A119,pp.172-81,1928.

[7]Grefenstette, J.J. 1996 Optimization Of Control Parameters For Genetic Algorithms. IEEE-SMC,SMC-16,122-128.

[8]Goldberg, D. E. (1989c). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison Wesley.

[9]Haddad, S. Chang, S. Swaminathan, and J. Lien, "Degradation due to hole trapping in flash memory cells," *IEEE Electron Device Lett.,* vol. 10, pp. 1 17-1 19, Mar. 1989.

[10]Haddad, S. Chang, S. Wang, A. J. Bustillo, J. Lien, T. Montalvo, and M. Van Buskirk, "An investigation of erase-mode dependent hole trapping in Flash EEPROM memory cell," *IEEE Electron Device Lett.*, vol. 11,pp. 514–516, 1990.

[11]IEEE Std 1005-1998. "IEEE Standard Definitions And Characterization Of Floating Gate Semiconductor Arrays," New York: The Institute of Electrical and Electronic Engineers, Inc.

[12]Luger, G. Artificial Intelligence structures and strategies for complex problem solving. Addison Wesley , Fourth edition ISBN 0-201-64866-0

[13]Mohammad, M. K. Saluja, and A. Yap, "Testing Flash Memories," 13*th International Conference on VLSI Design*, pp. 406–411, 2000.

[14]Pavan, P. Bez, R. Olivo, P. and E. Zanomi. Flash memory cells-An overview. Proc.IEEE,85(8);1248-1271, Aug. 1997.

[15]Reeves, Colin R., "Using Genetic Algorithms With small population," Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo CA.

[16]Sakui, K. Ohuchi, and F. Masuoka, "Extended data retention characteristics after more than 10‘ write and erase cycles in EEPROMs," in *IEEE IRPS 1990,* pp. 259-264.

[17]Silicon Storage Technology, Inc "Technical Comparison of Floating Gate Reprogrammable Nonvolatile Memories" Silicon Storage Technology, Inc. 1171 Sonora Court, Sunnyvale, www.ssti.com

[18]Tam, S., et al., "Lucky Electron Model of Channel Hot-Electron Injection in MOSFETs", *IEEE Transactions on Electron Devices*, vol. ED-31, pp. 1116–1125, 1984.