# Analysis of Hot-Carrier AC Lifetime Model for MOSFET

by

Beniyam Menberu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 15, 1996



Author .....
Department of Electrical Engineering and Computer Science
January 29, 1996

Certified by .....
James Chung
Associate Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
F. R. Morgenthaler
Chairman. Department Committee on Graduate Theses

# ACKNOWLEDGMENT

I would like to thank my research advisor, Professor James Chung, for his support throughout the span of my research work. I am also thankful for his guidance that has made a difference on me. I am very proud to have worked for one of the most hard working and intelligent person I have known.

I would also like to thank my fellow research members Abraham Kim, Wenjie Jiang, Huy Le, Brian Stine, Jee-Hoon Y. Krska, Jung U. Yoon, and Rajesh Divecha. I am grateful for all the support and guidance I received from Abraham Kim and Wenjie Jiang. I would also like to appreciate Brian, Huy, Jee-Hoon, Jung and Rajesh for the help they have given me throughout my research.

My father, Menberu Yemata, has always been the closest person to me throughout my life. I would like to thank him for all the love he has given to me. My only wish was that my mother would have been here to have seen my accomplishments. May her soul rest in peace. I am also grateful for my brother Dawit, and my sisters Hiwot and Kidist for their support.

# Analysis of Hot-Carrier AC Lifetime Model for MOSFET

by

Beniyam Menberu

Submitted to the
Department of Electrical Engineering and Computer Science

on January 29, 1996

In partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

## Abstract

This study presents a new algorithm for the prediction of AC hot-carrier lifetime/degradation of NMOSFETs. The hot-carrier degradation model parameters used in this study were modeled as a quadratic function of the oxide electric field. The new algorithm models the non-linear behavior of the AC hot-carrier degradation accurately using the method of the dominant degradation asymptote. The formulation of the dominant degradation asymptote will be described. Circuit examples are shown to illustrate that the new algorithm is better than the current method in predicting AC hot-carrier degradation. The current method is also shown to overestimate AC lifetime significantly compared to the new algorithm.

Thesis Supervisor: James Chung
Title: Associate Professor of Electrical Engineering

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Overview of Hot-Carrier Effects

Hot-carrier degradation is the result of physical mechanisms that are present in MOSFETs undergoing voltage stress. The electric field at the drain side, $E_d$, is the driving force for hot-carrier degradation. Because of the high electric field at the drain side, the mobile electrons in the conducting channel of a MOSFET gain high energy as they approach the drain side. These electrons with high energy can break Si-H bonds near the $Si/SiO_2$ interface at the drain side which can result in the outward diffusion of hydrogen from the interface[1]. This bond breaking can result in the generation of interface traps that can have a detrimental effect on MOSFET performance.

As the dimensions of MOSFETs become smaller and smaller, the issue of hot-carrier reliability becomes a major concern for circuit designers. The presence of hot-carriers in MOSFETs results in device performance degradation. Interface trap generation $\Delta N_{it}$ results in change of the threshold voltage $\Delta V_T$, and reduction of the MOSFET drain current $\frac{\Delta I_d}{I_{do}}$ and transconductance $\Delta g_m$ [1]. These changes in device performance can affect overall circuit performance.

## 1.2 AC Hot-carrier Circuit Degradation

For AC waveforms, the quasi-static approximation is used to predict the MOSFET degradation/lifetime. The AC waveforms are partitioned in time by small time-steps such that DC conditions apply within each time-step. The quasi-static approximation enables the use of the DC degradation model within each timestep in order to predict AC degradation.

The DC hot-carrier degradation model is defined by three parameters **m,n,H.** The parameter **n** is the degradation rate. The parameter **m** is the voltage acceleration factor and **H** is a process dependent constant. These model parameters can be extracted from drain current reduction $(\frac{\Delta I_d}{I_{do}})$ measurements and the corresponding lifetime $\tau$ of MOSFETs stressed at different DC bias voltages. The quasi-static approximation predicts AC degradation accurately if the AC waveform does not change too rapidly.

Using the quasi-static approximation, the AC lifetime can be calculated at a future time point using an iterative set of equations. However the drawback in using the iterative equations is that it can become computationally infeasible to calculate AC lifetime if the time point of interest is much greater than the period of the AC waveform. For example, it may take years to calculate the AC degradation of a device stressed for only several seconds of AC operation (which corresponds to billions of waveform cycles) using the iterative equations. Therefore there is a need for an accurate AC hot-carrier degradation model that is not CPU-time extensive.

## 1.3 Current Method for AC Hot-carrier Degradation

The current AC lifetime prediction method used by BERT(Berkeley Reliability Tools) predicts AC degradation by extrapolation from the first few simulated data points. These simulation data points are obtained by using the exact AC iterative equation. This current $\mathbf{n_{eff}}$ methodology is shown in Figure 1.1. Simulation results from the first two AC waveform cycles are used to determine the AC degradation model parameters $\mathbf{n_{eff}}$ and $\mathbf{A_{eff}}$ that define the line shown in Figure 1.1. The AC degradation at a future time point can then be found by extrapolating the line to the future time point of interest.

Usually the future time-point of interest is on the order of $10^{16}$ cycles and using only the first two cycles to extrapolate to $10^{16}$ cycles can result in huge extrapolation errors.

This extrapolation error is introduced because the AC degradation can exhibit a complex power-law time dependence.

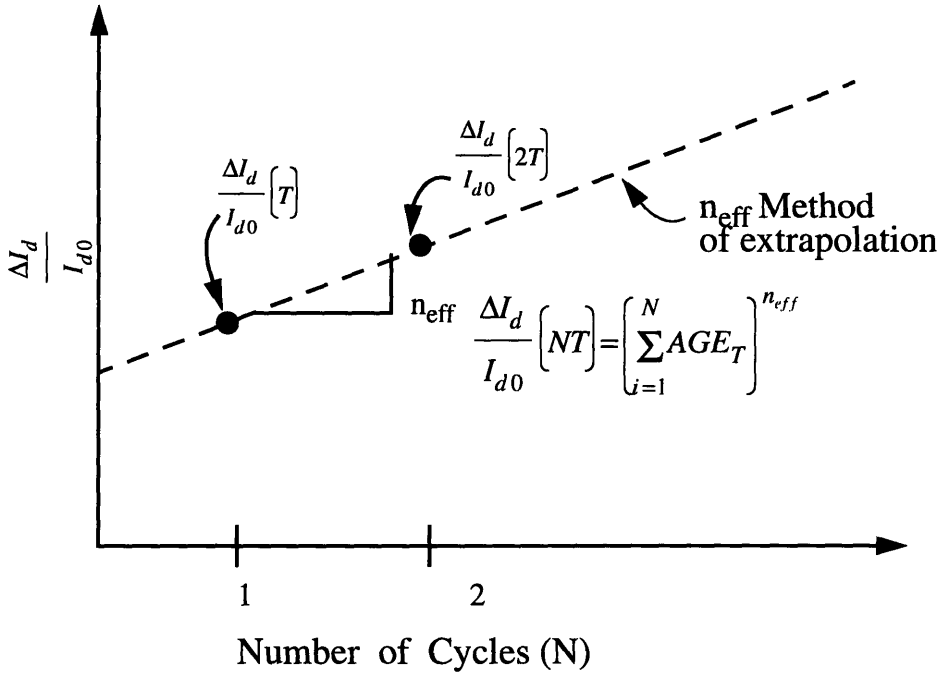


**Figure 1.1** The current $\mathbf{n_{eff}}$ method of calculating AC degradation.

## 1.4 Proposed New Method for AC Hot-carrier Degradation

For typical digital circuits, $\mathbf{V_{gd}}$ (gate-to-drain bias voltage) of the pull-down NMOS transistor can vary from $\pm V_{dd}$ to 0 during transitions of the input signal. Since the degradation rate **n** has significant bias dependency on the oxide-electric field $E_{ox} \propto V_{gd}$ [4], modeling the degradation with just a single degradation rate $\mathbf{n=n_{eff}}$(such as used in the current $\mathbf{n_{eff}}$ method) can lead to significant underestimation of the AC degradation or overestimation of the AC lifetime. Therefore, the proposed new method models AC degradation using more than a single degradation rate **n**. The new method uses the concept of the "dominant" degradation asymptote. Using the quasi-static approximation, the degradation rate **n** can be calculated at each time step as a function of the drain and gate voltage. A

series of averaged values of **n**(weighted by the instantaneous age) and the AGE(the cumulative value of the instantaneous AGE) calculated from the NMOS voltage waveforms are used to determine the AC degradation asymptote. The new method uses these AC degradation asymptotes to give an accurate lower-bound on the AC degradation at any specified future time-point.

In Figure 1.2, the method of the dominant degradation asymptote is demonstrated schematically. The true AC degradation(determined by the AC iterative equation) can be approximated as a "superposition" of different DC asymptotes. At the particular time point of interest, which in this case is 10 years, the asymptote that predicts the highest degradation has to be identified. This asymptote is defined as the dominant asymptote. An initial asymptote in Figure 1.2 is defined by two variables $AGE_T$ and $\mathbf{n_{ave}}$ where $\mathbf{n_{ave}}$ is the average **n** weighted by the instantaneous age. The asymptotes labeled B & C from Figure 1.2 are then generated from the asymptote labeled A by using $\mathbf{n_{ave}}$ to partition the initial asymptote A. According to Figure 1.2, asymptote B is more dominant than asymptote A at 10 years since it predicts higher degradation. This asymptote generation process is continued until the most dominant asymptote has been identified.

In terms of computational efficiency, both the new method and the current $\mathbf{n_{eff}}$ method require significantly less CPU time than the iterative method of predicting AC degradation. However the new method of lifetime prediction, based on dominant asymptotes, takes into account the dynamic behavior of the degradation equation. The new method adjusts to the behavior of the AC degradation by finding the dominant asymptote that accurately bounds the AC degradation at the specified time point. In contrast, only a single degradation rate **n** is used to predict AC degradation for all times with the current $\mathbf{n_{eff}}$ methodology[2]. Thus the new method can predict AC degradation with more accuracy.

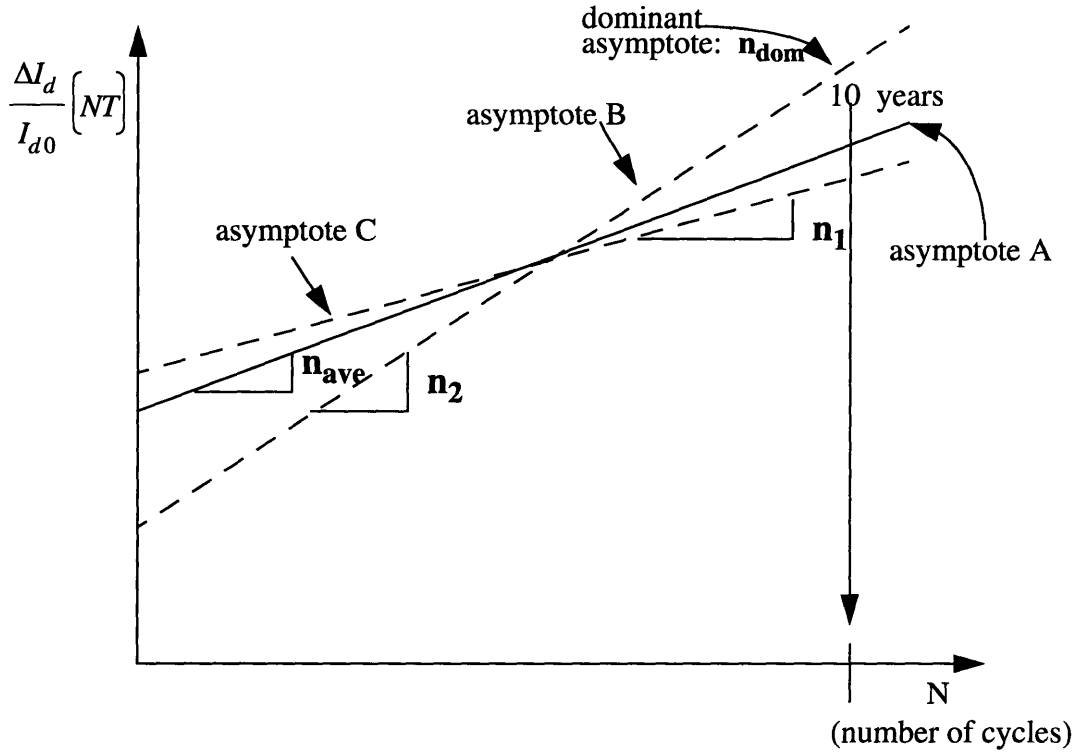


**Figure 1.2** Determining AC degradation based on identifying the dominant degradation asymptote.

# Chapter 2

# DC Hot-carrier Degradation Model

## 2.1 The Substrate Current Model

Hot-carrier degradation in NMOSFET devices has been shown to correlate very well with substrate current, $I_{sub}$[2]. Thus $I_{sub}$ can be used as a good monitor for device-level degradation due to hot-carriers. The generation of $I_{sub}$ is the result of the lateral electric field present at the drain side of the MOSFET [2]. Electrons along the channel gain high enough energy from the lateral electric field to cause impact ionization near the drain. The impact ionization results in the formation of electron-hole pairs. The substrate current is the hole current generated when the holes created by the impact ionization are attracted toward the substrate by the repelling electric field. A more complete model of the substrate current generation is shown in Figure 2.1 [2] with different arrows representing hole and electron currents. The hole currents in Figure 2.1 make up the substrate current.

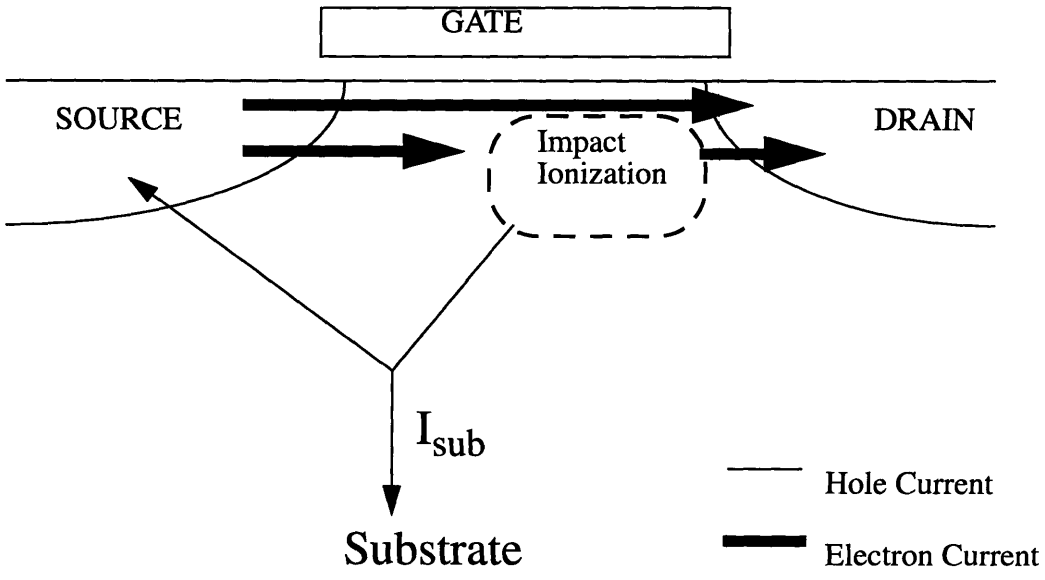


**Figure 2.1:** Substrate current generation in MOSFETs

The substrate current ($I_{sub}$) is a function of the drain current $I_d$ and a few other parameters that can be extracted from drain and substrate current data. A general equation for the hot-electron induced substrate current is[1]:

$$I_{sub} = CI_d e^{-\frac{\varphi_i}{q\lambda E_m}} \tag{2.1}$$

where C is a process-dependent parameter, the variable $\varphi_i$ is the critical energy for impact ionization, $\lambda$ is the mean free path for electrons, and $E_m$ is the maximum electric field at the drain[1]. The substrate current model is[3]:

$$I_{sub} = \left(\frac{A_i}{B_i}\right) I_d (V_{ds} - V_{dsat}) e^{\left(\frac{-B_i l_c}{V_{ds} - V_{dsat}}\right)} \tag{2.2}$$

where $V_{ds}$ is the drain to source voltage and $V_{dsat}$ is as follows[3]:

$$V_{dsat} = \frac{E_{crit} L (V_{gs} - V_T)}{E_{crit} L + (V_{gs} - V_T)} \tag{2.3}$$

The parameters $A_i$ and $B_i$ are the impact ionization coefficients, $V_{dsat}$ is the drain saturation voltage, and $V_T$ is the threshold voltage. For long channel MOSFETs, the $V_{dsat}$ reduces to the familiar equation $V_{dsat} = V_{gs}$-$V_T$. The parameter $l_c$, which is the length of the "effective pinch-off" region [3] of the channel, can be approximated as follows[9]:

$$l_c = \left(\frac{t_{ox} x_j}{3}\right)^{\frac{1}{2}} \tag{2.4}$$

where $t_{ox}$ is the gate-oxide thickness and $x_j$ is the junction depth of the MOSFET. $E_{crit}$ is the critical field for velocity saturation[3].

## 2.2 Hot-electron Generation in MOSFETs

Hot-electron degradation is caused by electrons which have high enough energy to break Si-H bonds at the $Si/SiO_2$ interface[1]. This results in the generation of interface

traps at the $Si/SiO_2$ interface which can cause reduction in the MOSFET drain current, transconductance, and shifts in the MOSFET threshold voltage. Linear region drain current reduction, which is proportional to the amount of interface trap generation, is a good monitor for hot-electron-induced device-level degradation[3].

The equation that describes the creation of hot-electron-induced interface traps is as follows:[1]

$$\Delta N_{it} = C\left(\frac{I_d}{W} e^{-\frac{\varphi_{it}}{q\lambda E_m}} t_{stress}\right)^n \tag{2.5}$$

In Equation 2.5, C is a process-dependent constant. The variable $t_{stress}$ is the amount of time the device has been stressed. The variable $\varphi_{it}$ is the critical energy for interface-state generation. Substituting Equation 2.1 into 2.5, an equation for $\Delta N_{it}$ can be derived:

$$\Delta N_{it} = \left(\left(\frac{I_{sub}}{I_d}\right)^m \frac{I_d}{WH} t_{stress}\right)^n \tag{2.6}$$

where $\Delta N_{it}$ now is a function of quantities that can either be experimentally measured or simply calculated. The parameters **m,n,H** in Equation 2.6 model the interface state generation $\Delta N_{it}$. The parameter **m**, the voltage acceleration factor, is defined as:

$$m = \frac{\varphi_{it}}{\varphi_i} \tag{2.7}$$

The parameter **n** is the degradation rate and **H** is a process-dependent constant. Equation 2.6 also describes the behavior of such degradation monitors as the drain current reduction $\frac{\Delta I_d}{I_{do}}$, transconductance shift $\Delta g_m$, and threshold voltage shift $\Delta V_T$ which all linearly depend on $\Delta N_{it}$. For a DC stressed device, Equation 2.6 represents the exact amount of degradation.

## 2.3 Hot-electron AGE

Hot-electron AGE is a measure of how much a device has been stressed. The variable AGE can be defined in terms of an integral as follows:

$$AGE\left(t_{stress}\right) = \int_0^{t_{stress}} \left(\frac{I_{sub}}{I_d}\right)^m \frac{I_d}{WH} dt = \int_0^{t_{stress}} A\,(t)\,dt \tag{2.8}$$

where *A(t)*, the instantaneous AGE, is defined as follows:

$$A\,(t) = \left(\frac{I_{sub}\,(t)}{I_d\,(t)}\right)^m \frac{I_d\,(t)}{WH} \tag{2.9}$$

In general, the instantaneous AGE, the integrand of Equation 2.8, defined as *A(t)* in Equation 2.9, is a function of time. Figure 2.2 illustrates the numerical integration of the function A(t) as defined in Equation 2.9. The x axis is subdivided into timesteps $\Delta t$ where the *i*th timestep is defined as $\Delta t_i$. In general the timesteps may not be uniform due to the variations of the voltage waveform. For increased accuracy, options in SPICE allow the timesteps to be smaller where the AC waveforms vary rapidly[8]. The y axis is the quantity A(t) defined in Equation 2.9. The area under the curve in Figure 2.2 from t=0 to t=T(period) is the AGE over one period of the waveform defined as $AGE_T$. Using the quasi-static approximation, the area within each timestep, determines the DC stress condition within that timestep.

The shaded region in Figure 2.2 represents the approximation to the exact AGE(t). The AGE within each timestep can be approximated by the area of the trapezoid shown in Figure 2.2. Using this approximation, the integration in Equation 2.8 over one period T can be done numerically using the trapezoidal method[7]:

$$AGE_T = \int_0^T A(t)\,dt = \sum_{i=0}^{M-1}\left(\frac{A_i + A_{i+1}}{2}\right)\cdot \Delta t_i \tag{2.10}$$

$$= (AGE_0 + AGE_1 + AGE_2 + \ldots + AGE_{M-1}) \tag{2.11}$$

In Equation 2.11, $AGE_i$ is the total age in the *i*th interval and M is the total number of timesteps in a period. Since the waveform is periodic, the AGE at any time t=NT is proportional to $AGE_T$.

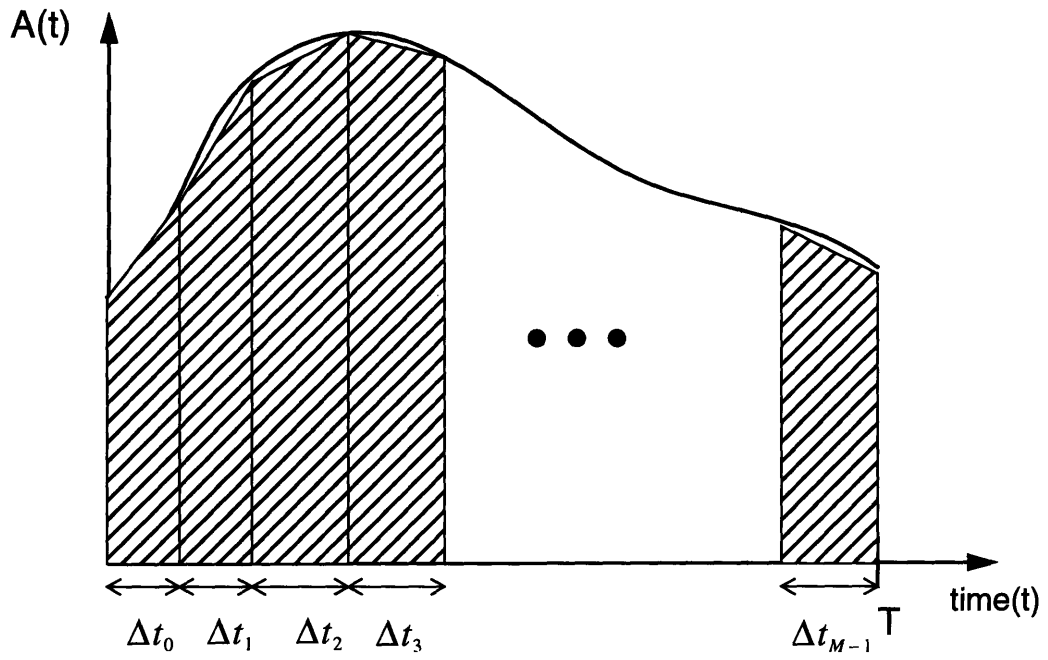


**Figure 2.2:** Numerical integration of the function A(t)

For a DC waveform, *A(t)* is a constant and comes out of the integral in Equation 2.8. Equation 2.8 can then be substituted in Equation (2.6) to define the general equation for degradation in DC stressed devices as follows:

$$\Delta N_{it} = (AGE)^n \tag{2.12}$$

In general, DC hot-electron degradation can be described by AGE and the parameter **n**.

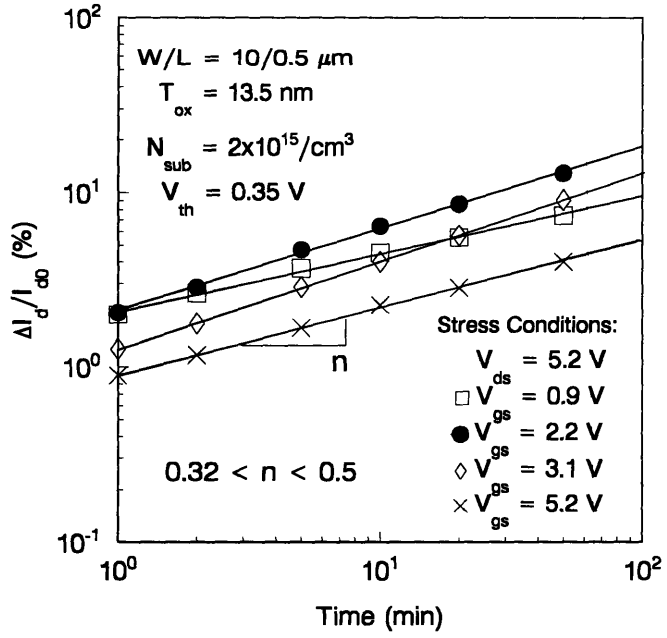


**Figure 2.3:** DC degradation displaying different **n** coefficients

## 2.4 Parameter Extraction for m,n,H

In order to accurately predict DC hot-carrier degradation, the parameters **m,n,H** and their bias dependencies have to be extracted precisely. The parameters **m,n,H** can be extracted from DC stress data. The degradation rate **n** can be extracted from the data shown in Figure 2.3[6]. The degradation rate **n** is just the slope of the line in Figure 2.3. The data in Figure 2.3 were obtained by DC stressing NMOS devices at particular bias conditions and monitoring the resulting degradation(drain current reduction) at different time intervals. The different lines in Figure 2.3 correspond to the varying stress conditions and different oxide electric field $E_{ox}$. The oxide electric field is defined in terms of the gate to drain voltage $V_{gd}=V_g-V_d$, the flat-band voltage $V_{fb}$, and the gate-oxide thickness $t_{ox}$:

$$E_{ox} = \frac{V_g - V_d - V_{fb}}{t_{ox}} \tag{2.13}$$

The $E_{ox}$ dependencies of the degradation rate **n** can be found by varying the stressing condition, hence $E_{ox}$, and measuring the degradation rate **n**.

The **m** and **H** parameters can be extracted by plotting the DC device lifetime versus the current ratio $\frac{I_{sub}}{I_d}$ [2]. If the device lifetime definition is $\Delta D_f$, then the lifetime can be defined as follows:

$$\tau = \Delta D_f^{\frac{1}{n}} \cdot \left(\frac{I_{sub}}{I_d}\right)^{-m} \cdot \frac{WH}{I_d} \tag{2.14}$$

Rearranging Equation 2.14 [2]:

$$\frac{\tau \cdot I_d}{W} = \Delta D_f^{\frac{1}{n}} \cdot \left(\frac{I_{sub}}{I_d}\right)^{-m} \cdot H \tag{2.15}$$

Equation 2.15 states that on a log-log plot of $\frac{\tau \cdot I_d}{W}$ versus $\frac{I_{sub}}{I_d}$, the slope will be -**m** and the intercept will be **H.**

In order to extract the parameters **m** and **H,** a set of devices has to be stressed at a series of different fixed $E_{ox}$. For a given current ratio $\frac{I_{sub}}{I_d}$, the device lifetime $\tau$ at a given lifetime definition has to be calculated. The current ratio and the lifetime determine a single data point shown in Figure 2.4 [4]. The other data points on a line, which corresponds to a fixed $E_{ox}$, are generated by varying $V_g$ and $V_d$ while keeping $E_{ox}$ fixed so as to change the stress current ratio. The lifetime is again calculated for the new current ratio and another point on the line is generated. The same process is repeated for the other lines in Figure 2.4 which correspond to different $E_{ox}$. From Figure 2.4 and Equation 2.15 the parameter -**m** is the slope of the line and **H** is the intercept.

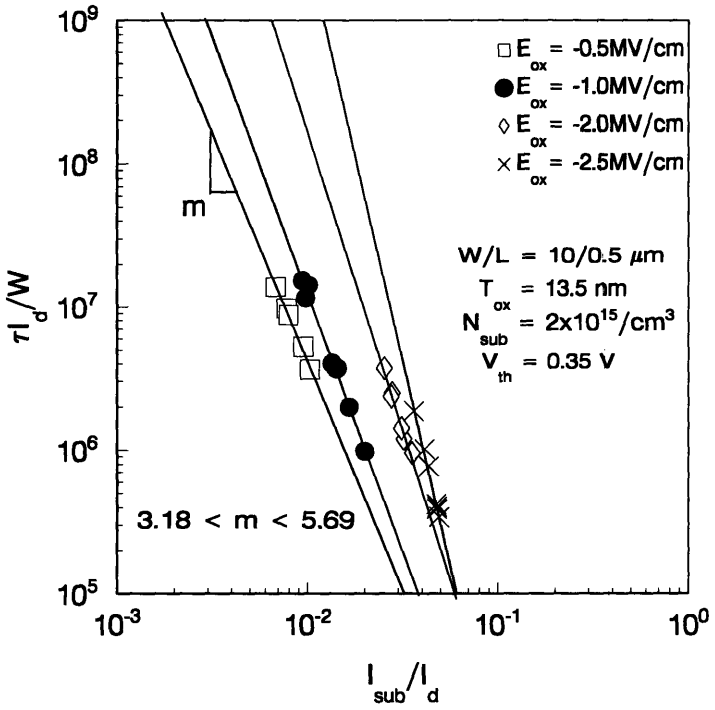


**Figure 2.4:** Extracting the parameters **m** and **H**

It has been shown that the degradation parameters **m,n,H** have a significant oxide electric field dependency $E_{ox}$[4]. Thus, it is necessary to characterize the $E_{ox}$ dependencies of the degradation parameters for predicting accurate AC degradation. The test devices used in the characterization of the $E_{ox}$ dependency are abrupt-junction NMOSFETs with $t_{ox}$ ranging from 9-19nm and $N_{sub}$ ranging from $1x10^{15}$ - $9x10^{15}$ and $L_{eff} > 0.4$ [4]. The stress voltage ($V_{gd} = V_g - V_d$) was varied to determine the $E_{ox}$ dependencies of the parameters **n** and **m**.

Figure 2.5 [6] shows the degradation parameters **n** and **m** as a function of $E_{ox}$. The parameters **n** and **m** can be modeled as a quadratic function of $E_{ox}$ as shown in Figure 2.5. The quadratic model for the parameters **n** and **m** is as follows:

$$n\left(E_{ox}\right) = -0.0239 \times \left(E_{ox}\right)^2 - 0.0747 \times E_{ox} + 0.4344 \tag{2.16}$$

$$m\left(E_{ox}\right) = 0.753 \times \left(E_{ox}\right)^2 + 1.144 \times E_{ox} + 3.703 \tag{2.17}$$

where $E_{ox}$ has units of MV/cm. The degradation rate **n** varies significantly ($0.32 < \mathbf{n} < 0.5$) over the range of stress voltages [4]. In the past, this bias dependency of **n** was not taken into account. The AC degradation is often predicted using either an average or peak **n** value. The current $\mathbf{n_{eff}}$ model for AC hot-carrier degradation does not model the $E_{ox}$ dependency accurately because it models **n** and **m** with a linear fit rather than the quadratic fit shown in Equation 2.16-17. Significant error can be introduced by using a linear fit for the parameters **n** and **m,** especially at low stress voltages where the $E_{ox}$ dependency is nearly flat.

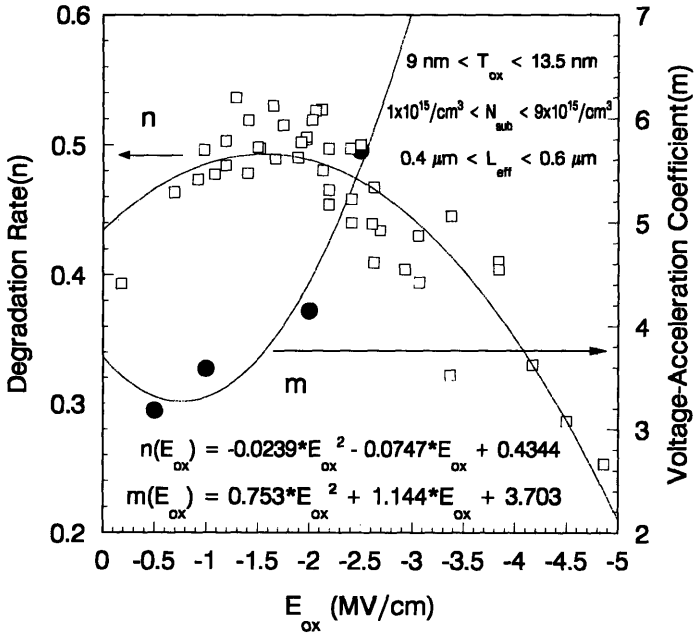


**Figure 2.5:** Bias dependencies of degradation parameters **m, n**

# Chapter 3

# AC Hot-carrier Degradation Model

## 3.1 AC Hot-carrier Degradation Equations

The AC hot-carrier degradation equations are derived using the quasi-static approximation. In the quasi-static approximation, the AC waveform is partitioned into small timesteps such that DC stress conditions are applicable within each timestep. Thus the degradation within each timestep can be calculated using the DC degradation model. Therefore, the AC degradation equations can be defined in terms of DC degradation equations. The AC degradation equations are expressed as a set of iterative equations whereby the degradation at any timepoint is dependent on the degradation from the previous timestep

**Two Step**: For the two-step example, the quasi-static approximation can be used to predict AC hot-carrier degradation. The two-step example consists of two DC stress conditions $S_1$ and $S_2$(i.e. $V_{gd1}$ ($0 < t < T_1$) and $V_{gd2}$ ($T_1 < t < T$) over a Period=T) that alternate within every cycle. The stress condition $S_1$ corresponds to degradation rate $\mathbf{n_1}$ and AGE=$AGE_1$ and the stress condition $S_2$ corresponds to degradation rate $\mathbf{n_2}$ and AGE=$AGE_2$.

The degradation ( $\Delta D$ )until $t = T_1$ can be calculated using Equations 2.8 and 2.12:

$$\Delta D\left(T_1\right) = \left( \int_0^{T_1} \left( \frac{I_{sub}}{I_d} \right)^m \frac{I_d}{WH} dt \right)^{n_1} = \left( AGE_1 \right)^{n_1} \tag{3.1}$$

Note, that the degradation at t=T is not simply the sum of the degradation due to each of the stressing conditions:

$$\Delta D\,(T) \neq \Delta D\,(T_1) + \left( \int_{T_1}^{T} \left( \frac{I_{sub}}{I_d} \right)^m \frac{I_d}{WH} dt \right)^{n_2} \neq (AGE_1)^{n_1} + (AGE_2)^{n_2} \tag{3.2}$$

Equation 3.2 implies that $\Delta D\,(T)$ is not a simple cumulative function of AGE when the degradation rate **n** varies with time. However, $\Delta D\,(T)$ can be calculated iteratively. For the two step case, $\Delta D\,(T)$ can be determined by finding an equivalent AGE=$AGE_{eq}$ such that the degradation resulting from this $AGE_{eq}$ with degradation rate $\mathbf{n_2}$ is equal to the degradation after t=$T_1$.

$$\left(AGE_{eq}\right)^{n_2} = \Delta D\left(T_1\right) = \left(AGE_1\right)^{n_1} \tag{3.3}$$

Solving for $AGE_{eq}$:

$$AGE_{eq} = \left(AGE_1\right)^{\frac{n_1}{n_2}} \tag{3.4}$$

Using the equivalent $AGE_{eq}$ notation, the **n** is now effectively constant (**n** = $\mathbf{n_2}$) throughout the whole period T since now $\Delta D\left(T_1\right)$ can be rewritten as follows:

$$\Delta D\left(T_1\right) = \left(AGE_{eq}\right)^{n_2} \tag{3.5}$$

If the degradation rate **n** is constant throughout the period T, the degradation at t=T is just the total age raised to the degradation rate **n**=$\mathbf{n_2}$. The total age is just the sum of $AGE_{eq}$ and $AGE_2$. Thus the degradation at t=T can be defined as follows:

$$\Delta D\,(T) = \left(AGE_{eq} + AGE_2\right)^{n_2} = \left(\left(AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{n_2} \tag{3.6}$$

Equation (3.6) represents the degradation at t=T using the iterative method for the two-step example. Equation 3.6 shows that the AC degradation for the two-step can be defined in terms of DC degradation quantities.

The same procedure is repeated when calculating the degradation at an arbitrary number of cycle N. Equation 3.6 can be extended for 2 cycles using the iterative procedure:

$$\Delta D\,(2T) = \left(\left(\left(\left(AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{\frac{n_2}{n_1}} + AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{n_2} \tag{3.7}$$

and in general the degradation for a two step waveform at t = NT is as follows:

$$\Delta D\,(NT) = \left(\left(\left[\Delta D\left((N-1)\,T\right)\right]^{\frac{1}{n_1}} + AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{n_2} \tag{3.8}$$

The definition of the degradation in Equation 3.8 is recursive because the degradation at t = NT depends on the result at t = (N-1)T which in turn depends on the result t = (N-2)T and so on. Thus to calculate the degradation at the time of interest, which is usually 10 years, this will correspond to $N \cong \frac{3.15x10^8}{T}$ cycles. For a period T=10ns, $N \cong 3.15x10^{16}$ cycles, which means that Equation 3.8 is iterated 2xN=$6.3x10^{16}$ times since there are two computations per cycle for the two step example. This iterative computation can take years of CPU time which is not feasible.

**General:** Equation 3.8 can be derived for an arbitrary AC waveform which is not necessarily a two-step waveform. For AC waveforms, the use of the DC quasi-static approximation limits the number of timesteps in a period of a waveform. If the waveforms are rapidly varying, then the quasi-static approximation requires that the timesteps be small. But if the waveforms are slowly varying, then the limitation on the timesteps is not as restrictive.

The degradation equation for an arbitrary waveform can be derived using arguments similar to those in Equation 3.8. The arguments used in the two-step waveforms can now be generalized for this arbitrary waveform. The degradation after t= $\Delta t_0$ is just the $AGE_0$ raised to the degradation rate $\mathbf{n_0}$ in the first interval of the waveform.

The degradation after $t = \Delta t_0 + \Delta t_1$ according to Equation 3.6 is:

$$\Delta D\left(\Delta t_0 + \Delta t_1\right) = \left(\left(AGE_0\right)^{\frac{n_o}{n_1}} + AGE_1\right)^{n_1} \tag{3.9}$$

Iterating this equation at $t = \Delta t_0 + \Delta t_1 + \Delta t_2$ the degradation is:

$$\Delta D\left(\Delta t_0 + \Delta t_1 + \Delta t_2\right) = \left(\left(\left(AGE_0\right)^{\frac{n_o}{n_1}} + AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{n_2} \tag{3.10}$$

Thus Equation 3.10 can be iterated until the desired time. Iterating further, the degradation at t=T (the degradation after one waveform cycle) can be derived:

$$\Delta D\,(T) = \left(\left(\left(\left(AGE_0\right)^{\frac{n_o}{n_1}} + AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{\frac{n_2}{n_3}} + \ldots + AGE_{M-2}\right)^{\frac{n_{m-2}}{n_{M-1}}} + AGE_{M-1}\Bigg)^{n_{M-1}} \tag{3.11}$$

Again, an equation can be derived for multiple cycles (i.e. N>1).

$$\Delta D\,(NT) = \left(\left(\left(\left(\left(\Delta D\,[\,(N-1)\,T]\,\right)^{\frac{1}{n_o}} + AGE_0\right)^{\frac{n_o}{n_1}} + AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{\frac{n_2}{n_3}} + \ldots + AGE_{M-2}\Bigg)^{\frac{n_{m-2}}{n_{M-1}}} + AGE_{M-1}\Bigg)^{n_{M-1}} \tag{3.12}$$

As in the two-step waveform, calculating the degradation using the iterative Equation 3.12 is not feasible when N is large. For practical purposes, approximations have to be used to predict the degradation at t = NT with minimal error.

## 3.2 Two-step and AC Circuit Simulation Examples

**Two-step:** The two-step waveform example is simulated to show the degradation behavior of Equation 3.8. This equation is non-linear because within every cycle the degradation rate **n** changes as the stressing condition alternates. Simulated data will be shown to illustrate this non-linear behavior. For the two-step waveform there are two DC stress conditions in one period. The two DC stress conditions for the simulations are outlined in

Figure 3.1. The degradation rate **n** and the voltage acceleration factor **m** for each stress condition are calculated using Equation 2.16 and 2.17 using the respective $E_{ox}$ value. The values of $E_{ox}$ were chosen to make the two **n** values ($\mathbf{n_1}$ and $\mathbf{n_2}$) differ significantly.

The parameters in Figure 3.1 were used to simulate degradation using Equation 3.8. To illustrate the non-linearity of the equation, the degradation was simulated for few cycles. The simulation result is shown below in Figure 3.2. The two lines in Figure 3.2 represent the DC degradation asymptotes determined by the AGE and **n** parameters within each stressing condition. The x axis is the number of cycles. For the simulation the period T=10ns was subdivided into timesteps of 0.1ns such that there are 100 simulation points in one cycle and up to 1000 simulation points for 10 cycles.

From Figure 3.2, it is evident that the degradation is linear until a tenth of a cycle(t=0.1T) where a new stress condition is present. The degradation then flattens out

Period : T=10ns

| DC Period #1 | DC Period #2 |
|---|---|
| $E_{ox1}$ = -4.11 Mv/cm | $E_{ox2}$ = -1.56 Mv/cm |
| $n_1 = 0.33$ | $n_2 = 0.49$ |
| $T_1 = 0.1\ T = 1ns$ | $T_2 = 0.9\ T = 9ns$ |
| $AGE_1=\left[\frac{I_{sub_1}}{I_{d1}}\right]^{m_1}\frac{I_{d1}}{WH_1}\times T_1$ = 1.45e-23 | $AGE_2=\left[\frac{I_{sub_2}}{I_{d2}}\right]^{m_2}\frac{I_{d2}}{WH_2}\times T_2$ = 7.2e-18 |

**Figure 3.1:** DC stress conditions for the two-step waveform

when the new stress condition occurs because its contribution to the degradation is small initially. It is evident that the simulation data obeys an asymptotic behavior because it

tracks a single asymptote after only the first few cycles.

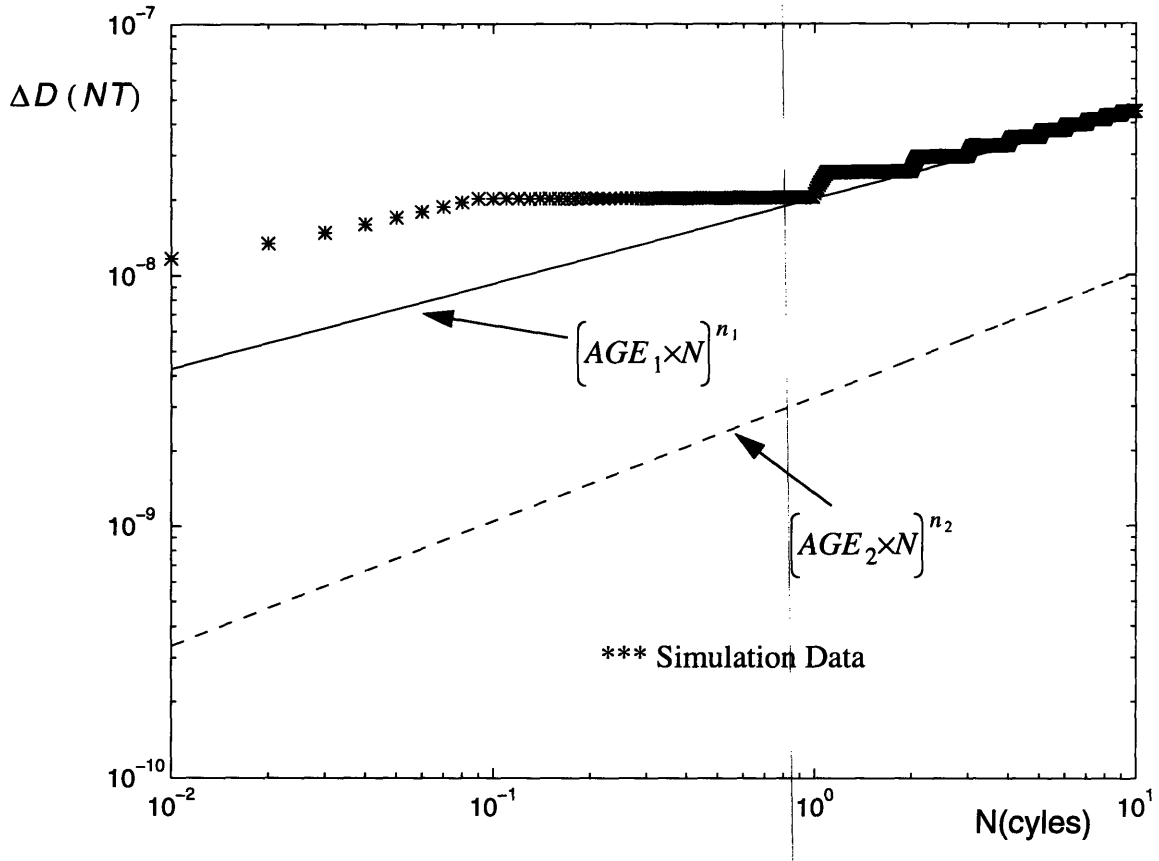


**Figure 3.2:** Degradation simulation for two-step waveform

The non-linearity of the degradation equation for the two-step example is also observed for large number of cycles. The same simulation for the two-step example was done up to N=3x$10^7$ cycles to illustrate this point. Figure 3.3 shows the two-step simulation data and the two DC degradation asymptotes shown in Figure 3.2. The degradation predicted by the current $\mathbf{n_{eff}}$ method is also shown. The intersection point of the two asymptotes is labeled $N_{int}$. The simulated data in Figure 3.3 tends to track the highest asymptote at any time. Thus the simulated data exhibits an asymptotic behavior for the two-step example.

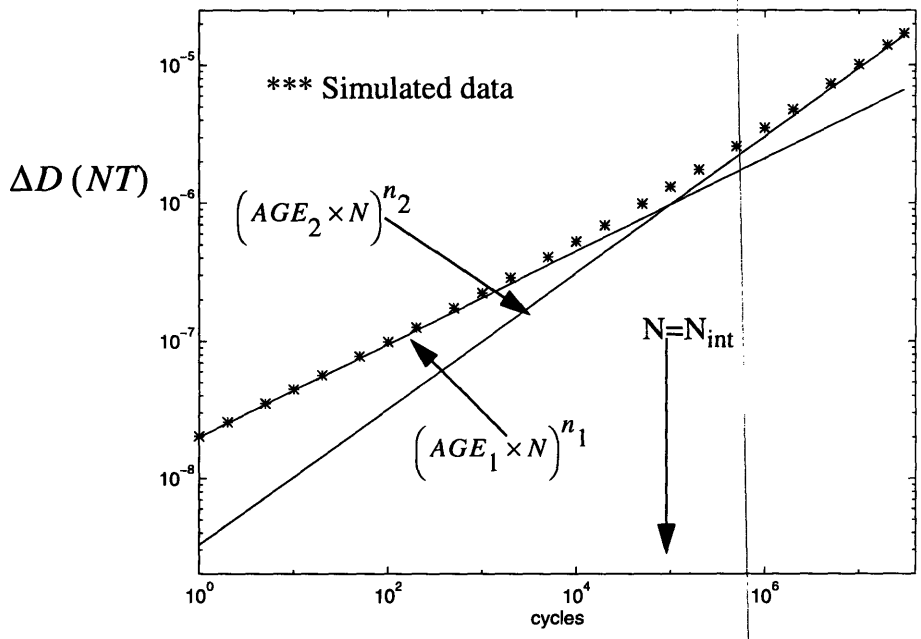


**Figure 3.3:** Asymptotic behavior of the two-step degradation equation.

**Inverter:** A similar simulation was done for an AC circuit. For this example a one-stage inverter with a capacitative load was chosen. The circuit is shown in Figure 3.4. The parameter $\alpha$ (V/ns) is the ramp rate of the input waveform $V_{in}$. The period of the waveform is T=10ns and the width of the NMOS is $W_n$=40um. For this simulation the ramp rate $\alpha$=10V/ns, $C_L$=0.1pF, and $V_{DD}$=5v. The degradation of the NMOS device in this circuit was simulated for 10 cycles and the simulation result is shown in Figure 3.5. The lines in Figure 3.5 represent the DC asymptotes.

The simulated data for the inverter in Figure 3.5 tracks one of the asymptotes just like the two-step simulation example. It also exhibits a non-linear behavior similar to that of Figure 3.2.

The non-linear behavior can also be observed for long simulations. This is shown in Figure 3.6. The simulation result show that the degradation for an inverter under AC oper-

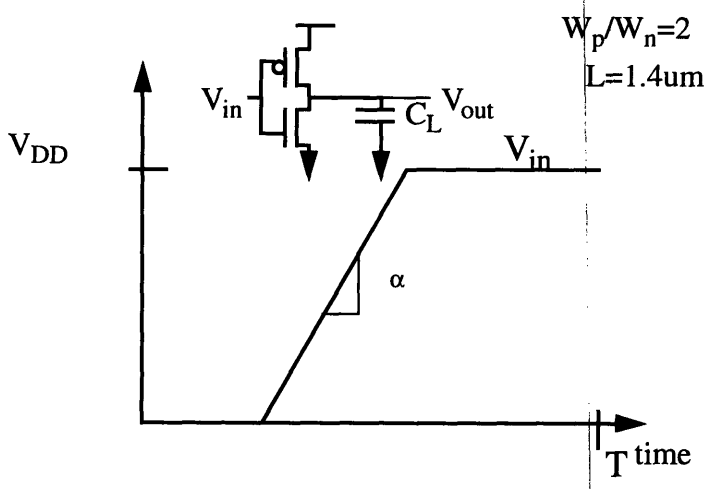


**Figure 3.4:** One-stage inverter with load $C_L$

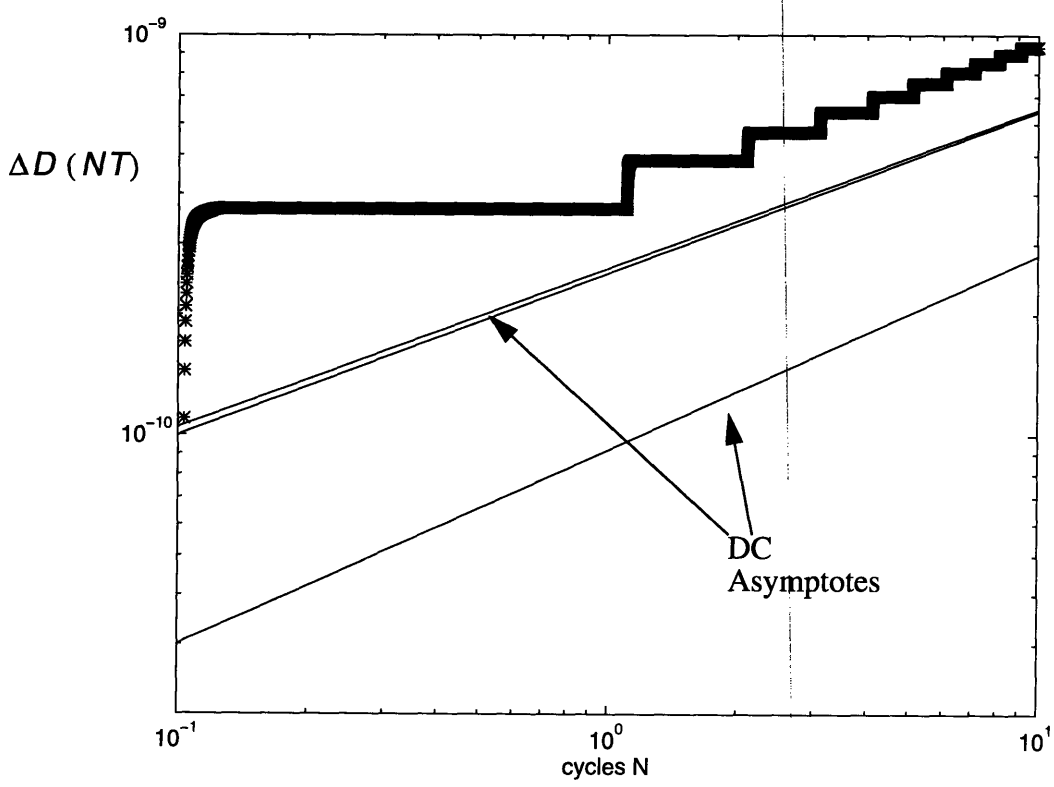


**Figure 3.5:** Degradation simulation for an inverter. ($W/C_L = 400\frac{\mu m}{pF}$, $\alpha = 10\frac{V}{ns}$)

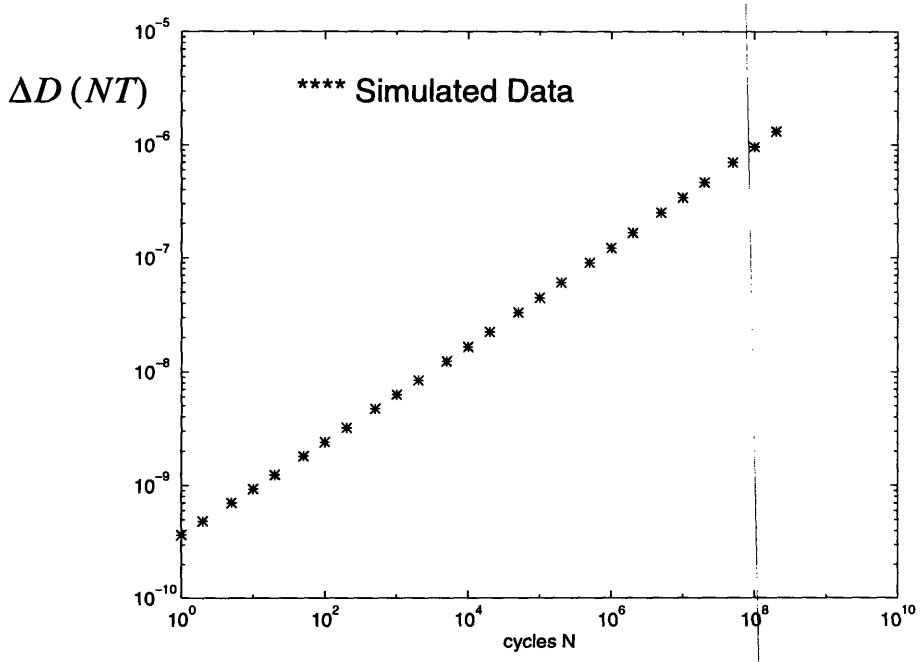


**Figure 3.6:** Degradation simulation for inverter(N=2x$10^8$ cycles)

ating condition behaves like the DC degradation data shown in Figure 2.3. Careful analysis of this simulated data will show that the degradation rate varies over time.

## 3.3 Current AC Hot Carrier Degradation Model($n_{eff}$)

The current $\mathbf{n_{eff}}$ AC hot-carrier degradation model is based on extrapolation from the first two waveform cycles. The exact degradation for the MOSFET is calculated iteratively using Equation 3.12 for the first two cycles(i.e. N=1,2). These two data points are used to fit the current $\mathbf{n_{eff}}$ model which is [2]:

$$\Delta D\,(t) = A_{eff} \times (t)^{n_{eff}} \tag{3.13}$$

The two fitting parameters are $A_{eff}$ and $\mathbf{n_{eff}}$. Since there are two parameters, two data points are needed to solve for them.

The data from the first two cycles of an arbitrary waveform can be used to solve for the fitting parameters $A_{eff}$ and $\mathbf{n_{eff}}$. Let $\Delta D\,(T)$ and $\Delta D\,(2T)$ be the exact degradation from the first two cycles of an arbitrary waveform. The two fitting parameters can be solved from two equations with two unknowns as follows:

$$n_{eff} = \frac{\ln\left(\frac{\Delta D\,(2T)}{\Delta D\,(T)}\right)}{\ln\,(2)} \tag{3.14}$$

$$A_{eff} = \frac{\Delta D\,(T)}{T^{n_{eff}}} \tag{3.15}$$

The current $\mathbf{n_{eff}}$ model uses these two parameters in Equations 3.14-15 to derive the degradation at any other time.

**Two Step:** For the two step waveform with the stress condition shown in Figure 3.1, Equations 3.14 and 3.15 will reduce to the following using Equation 3.6-3.7:

$$n_{eff} = \frac{\ln\left(\frac{\left(\left(\left(AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{\frac{n_2}{n_1}} + AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{n_2}}{\left(\left(AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{n_2}}\right)}{\ln\,(2)} \tag{3.16}$$

$$A_{eff} = \frac{\Delta D\,(T)}{T^{n_{eff}}} \tag{3.17}$$

For the set of stressing conditions in Figure 3.1, the $\mathbf{n_{eff}}$ equation can be simplified to:

$$n_{eff} \cong \frac{\ln\left(\frac{\left(\left(\left(2\times AGE_1\right)^{\frac{n_1}{n_2}}\right) + AGE_2\right)^{n_2}}{\left(AGE_1\right)^{n_1}}\right)}{\ln\,(2)} \cong \frac{\ln\left(\frac{\left(2\times AGE_1\right)^{n_1}}{\left(AGE_1\right)^{n_1}}\right)}{\ln\,(2)} = n_1 \tag{3.18}$$

$$A_{eff} = \frac{\left(\left(AGE_1\right)^{\frac{n_1}{n_2}} + AGE_2\right)^{n_2}}{T^{n_1}} \cong \left(\frac{AGE_1}{T}\right)^{n_1} \tag{3.19}$$

since $\left(AGE_1\right)^{\frac{n_1}{n_2}} \gg AGE_2$. Using Equation 3.13 and 3.18-19, the current $\mathbf{n_{eff}}$ model's equation can be rewritten:

$$\Delta D(NT) = A_{eff} \cdot (t)^{n_{eff}}\Big|_{t=NT} \cong \left(\frac{AGE_1}{T}\right)^{n_1} \cdot (NT)^{n_1} = \left(AGE_1 \cdot N\right)^{n_1} \tag{3.20}$$

This equation is just the DC asymptote for the two-step example shown in Figure 3.3. In Figure 3.3 the DC asymptote, represented by Equation 3.20, underestimates the degradation for $N > N_{int}$. Thus the current $\mathbf{n_{eff}}$ method underestimates the degradation for the two-step example. The error introduced by the extrapolation can be significant if the intersection point $N_{int}$ occurs very early in time.

**Inverter:** The main problem with the current $\mathbf{n_{eff}}$ model is the possible error that can result by extrapolating from just the first two cycles to 10 years(~$10^{16}$cycles) which is usually the time point of interest. The extrapolation error may seem small since the AC degradation appears to be linear in Figure 3.6 for the case of an inverter. However, this assumption is not correct because if a line was used to fit the simulated data in Figure 3.6, the slope of the line will change as more and more data points are used for the fitting.

To illustrate this point, a line was used to fit different numbers of data points from Figure 3.6. For each fit the slope of the line, **n**, was calculated to show that the AC degradation is non-linear. If the slope of the line did not change as more data points were used for the fitting, this would imply that the degradation was linear and that the degradation rate **n** was constant. However, for the inverter degradation in Figure 3.6, the slope of the line var-

ies as more data points are used to do the fitting. This implies that the AC degradation for the inverter is non-linear. The result of the fit is shown in Figure 3.7 where the x axis is the number of data points used in the fit and the y axis is the slope of the line that fits these data. As more data points are used to fit the line, the slope of the line(**n**) increases.

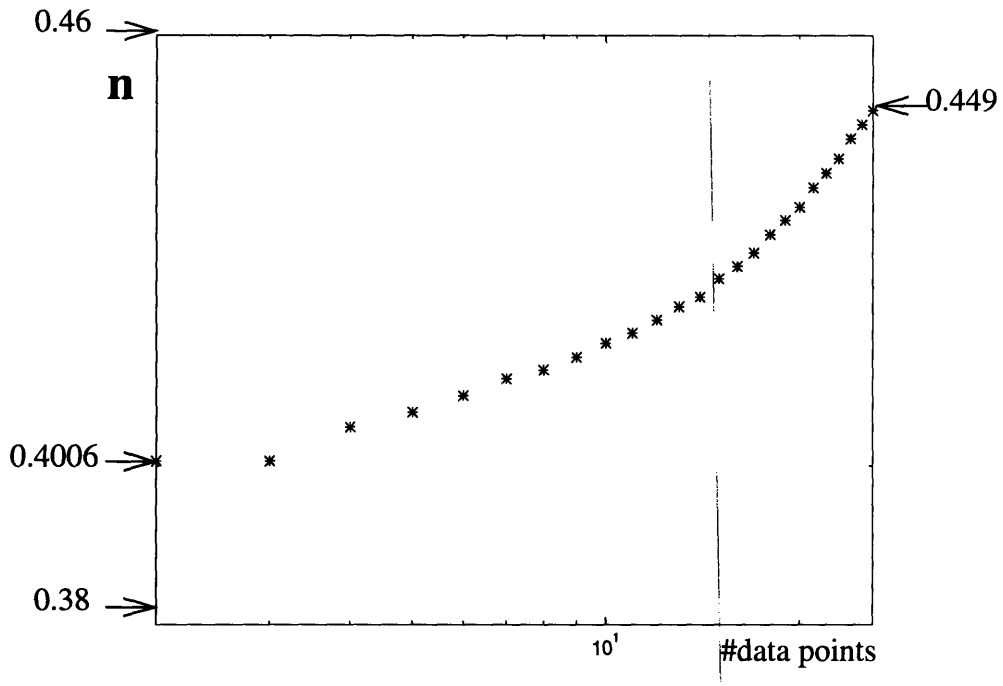


**Figure 3.7:** Degradation rate **n** as a function of number of data points used for fitting.

Thus by using only the data from the first two cycles(current $\mathbf{n_{eff}}$ method), the variations in the degradation rate **n** are not taken into account. The change in **n** is about ~0.05 according to Figure 3.7 for $N=2x10^8$ cycles but after $10^{16}$ cycles (~10 years) this change can become significant. Since the overall AC degradation is a sensitive function of **n**, small changes in **n** can have significant effect on the degradation. The degradation rate is dynamic and can have an appreciable change over many cycles and cannot be modeled accurately using only the first two cycles. Therefore different degradation rate **n**'s are needed to model degradation accurately depending on the time of interest.

**Fast Transients:** In special cases where the transition in the voltage waveforms($V_g$ and $V_d$) is very fast, the current $\mathbf{n_{eff}}$ method can actually predict the degradation accurately. Sharp transitions in the voltage waveforms will result in an impulse-like substrate current. A ring oscillator is a good example of a circuit where the substrate current exhibits such an impulse-like waveform. If the substrate current is impulse-like, the variable A(t) defined in Equation 2.9 will also be impulse-like. Equation 3.12 will reduce to an equation with only a single dominant degradation rate **n**. If the impulse occurs at the k*th* time interval in a period T, then the exact degradation at any time is approximately:

$$\Delta D(NT) \cong \left(AGE_k \times N\right)^{n_k} \tag{3.21}$$

Solving for $\mathbf{n_{eff}}$ and $A_{eff}$ using Equation 3.14 and 3.15:

$$n_{eff} = \frac{\ln\left(\frac{\Delta D(2T)}{\Delta D(T)}\right)}{\ln(2)} = \frac{n_k \times \ln\left(\frac{AGE_k \times 2}{AGE_k}\right)}{\ln(2)} = n_k \tag{3.22}$$

$$A_{eff} = \frac{\left(AGE_k\right)^{n_k}}{T^{n_{eff}}} = \left(\frac{AGE_k}{T}\right)^{n_k} \tag{3.23}$$

$$\Delta D(NT) = A_{eff} \cdot (NT)^{n_{eff}} = \left(\frac{AGE_k}{T}\right)^{n_k} \cdot (NT)^{n_k} = \left(AGE_k \times N\right)^{n_k} \tag{3.24}$$

Thus for impulse-like waveforms the current $\mathbf{n_{eff}}$ method can accurately model the degradation because only one **n** dominates. But for most circuits the substrate current waveform cannot be approximated as an impulse so that the current $\mathbf{n_{eff}}$ method has limited application.

# Chapter 4

# New Model for AC Hot-carrier Degradation

## 4.1 Concept of Dominant Asymptote

The AC degradation for both the two-step and the inverter example examined in Chapter 3 exhibited asymptotic behavior. It has been shown that the current $\mathbf{n_{eff}}$ method's lifetime predictions tend to diverge away from the true solution while an appropriately chosen asymptotic solution approaches the true solution. Therefore, the new model for AC hot-carrier degradation is based on the concept of a dominant asymptote. The new model's algorithm identifies the asymptote that accurately defines the AC degradation behavior of a device due to hot-carrier injection at the future time point of interest.

## 4.2 Formulation of the Initial Degradation Asymptote

An asymptote can be defined by two parameters AGE and **n** as follows:

$$\Delta D(NT) = (AGE \times N)^{n} \tag{4.1}$$

The first step in the algorithm for the new model is to generate an initial asymptote as a guess about the degradation at the time point of interest. The first asymptote is generated by calculating $AGE_T$ and **n** for the particular AC waveform. Assuming the AC waveform has a period T, then the variable $AGE_T$, the total AGE in a period, is defined as follows using Equation 2.8:

$$AGE_T = AGE(T) = \int_0^T \left(\frac{I_{sub}}{I_D}\right)^m \left(\frac{I_D}{WH}\right) dt \tag{4.2}$$

where the integrand is a function of time. The integration above is done numerically using the trapezoidal approximation shown in Figure 2.2.

Next, in Figure 2.2, an AGE=$AGE_i$ and degradation rate $\mathbf{n}=\mathbf{n}_i$ can be defined at the *i*th timestep, where *i* ranges from 0 to M-1, for each of the sub-intervals within the AC waveform. The degradation rate $\mathbf{n}=\bar{\mathbf{n}}$ of the initial asymptote is then defined as the average of all the $\mathbf{n}_i$'s in a waveform period weighted by their respective $AGE_i$'s. Thus, the average $\mathbf{n}=\bar{\mathbf{n}}$ is defined as follows:

$$\bar{n} = \frac{\sum_{0}^{M-1} n_i \cdot AGE_i}{AGE_T} \tag{4.3}$$

where $AGE_T$ in the denominator is defined in Equation 4.4 using Equation 2.11:

$$AGE_T = \sum_{0}^{M-1} AGE_i \tag{4.4}$$

This completes the specification for the initial asymptote. The equation of the initial asymptote is defined below:

$$\Delta D(NT) = \left(AGE_T \times N\right)^{\bar{n}} \tag{4.5}$$

Equation 4.5 will be used as a trial guess for the degradation at the time-point of interest.

## 4.3 Partitioning of the Asymptotes

The second step in the algorithm is to generate two asymptotes from the initial asymptote by partitioning the initial asymptote. The partitioning scheme involves several steps. For the illustration of these steps, an inverter example is used($W/C_L = 40\left(\frac{\mu m}{pF}\right)$, $\alpha = \frac{5}{7}\left(\frac{V}{ns}\right)$). The first of these steps is to generate a plot of **n** versus time for the period of the waveform. This is shown in Figure 4.1. In Figure 4.1, given **n** as a function of $E_{ox}$, **n** can be calculated at every timestep using the value of $V_{gd}$ (and hence $E_{ox}$) at that timestep. The second step is to plot the instantaneous AGE in each timestep versus time. This is shown in Figure 4.2. The next plot, shown in Figure 4.3, is to plot cumulative AGE versus

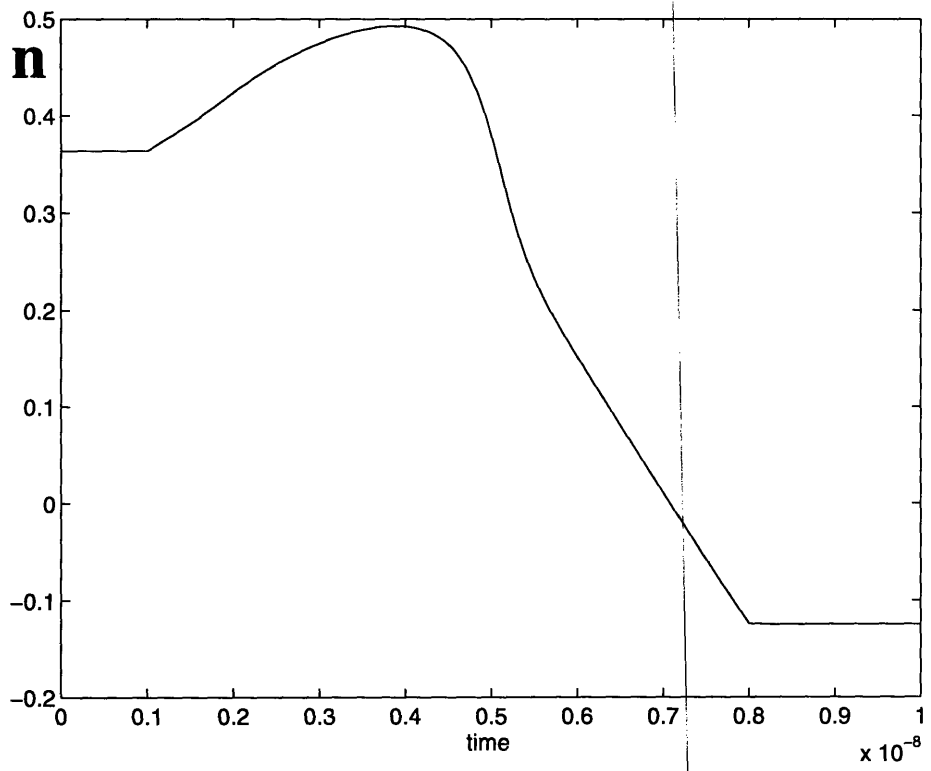


**Figure 4.1:** Degradation rate **n** vs. time(t)

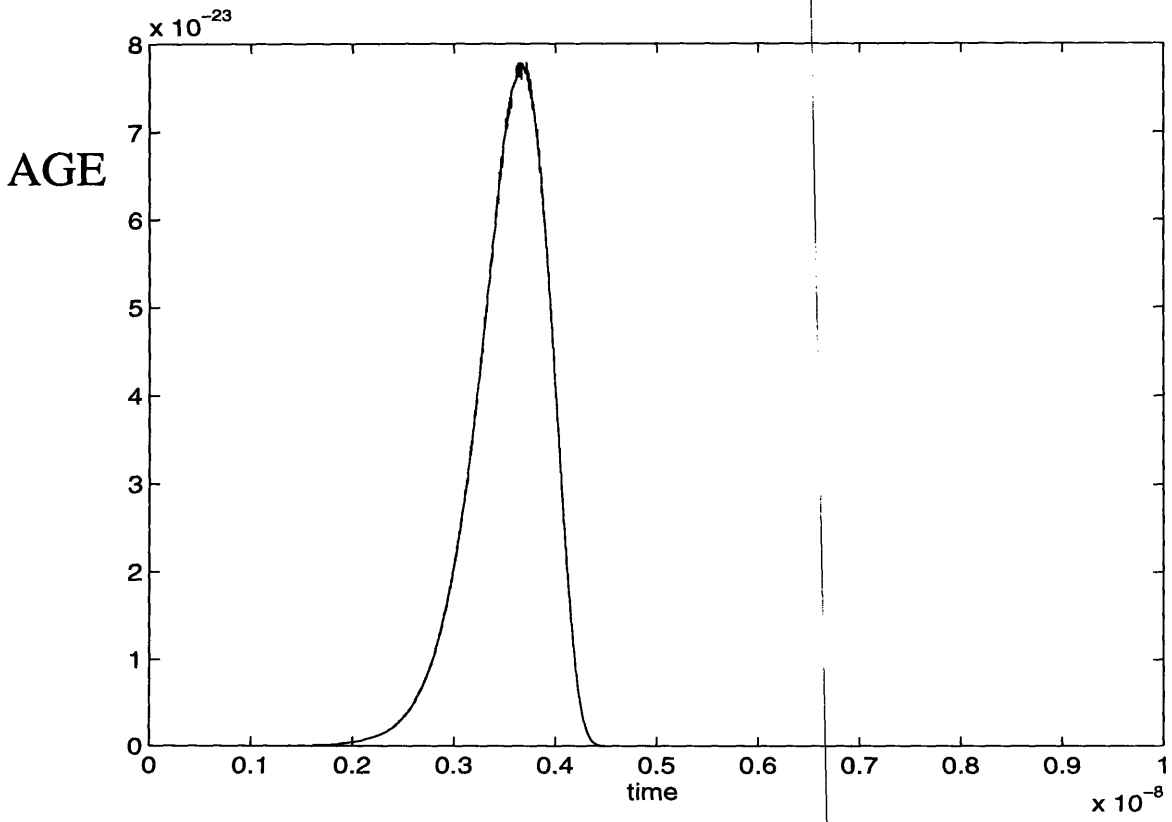


**Figure 4.2:** Instantaneous AGE versus time (t)

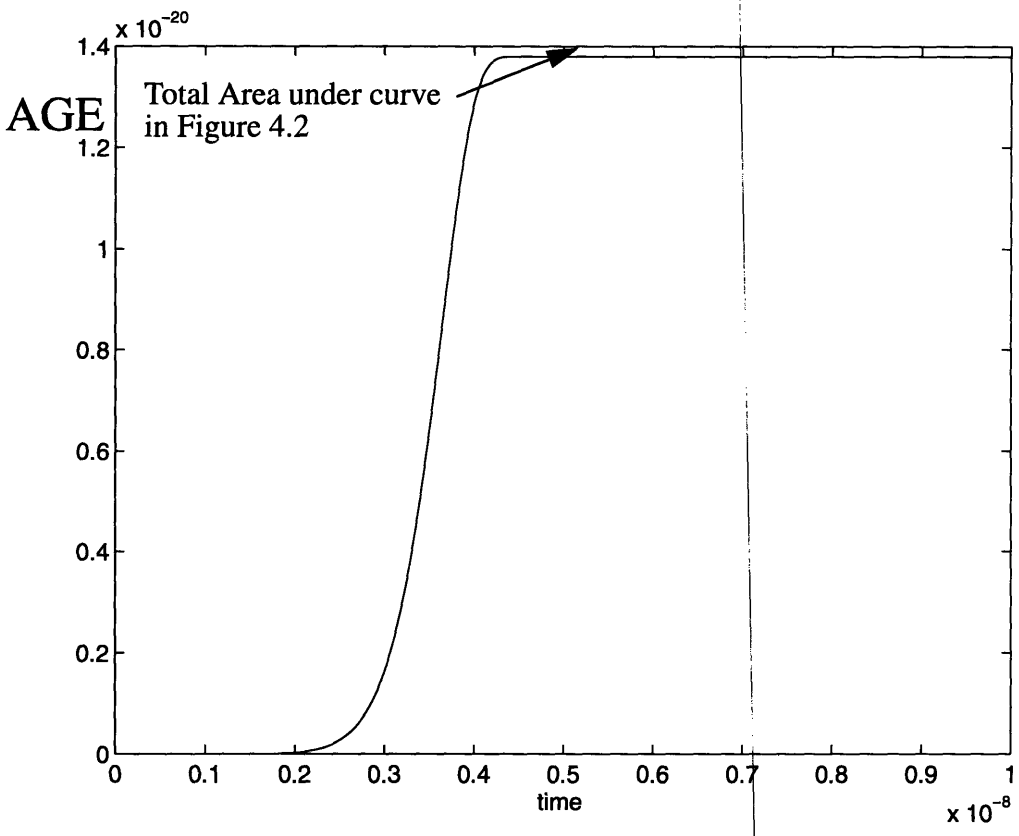


**Figure 4.3:** Cumulative AGE versus time(t)

time. Cumulative AGE at some time $t_o$ is defined to be the area under the curve in Figure 4.2 until time=$t_o$.

The degradation rate **n** versus cumulative AGE is shown in Figures 4.4 and 4.5. The x-axis is the cumulative AGE at some time t and the y-axis is the degradation rate **n** at the same time t. The horizontal lines shown in Figures 4.4 and 4.5 correspond to the average **n**. The line labeled $\bar{\mathbf{n}}$ is calculated using Equation 4.3.

This line defined by $\bar{\mathbf{n}}$ is then used to partition the initial asymptote into two components. The two components are defined by the parameters $AGE_1$, $\mathbf{n_1}$, $AGE_2$, and $\mathbf{n_2}$ shown

in Figures 4.4 and 4.5. The total $AGE_T$ is the sum of $AGE_1$ and $AGE_2$. The degradation rates $\mathbf{n_1}$ and $\mathbf{n_2}$ are the average of the **n**'s weighted by AGE below and above the

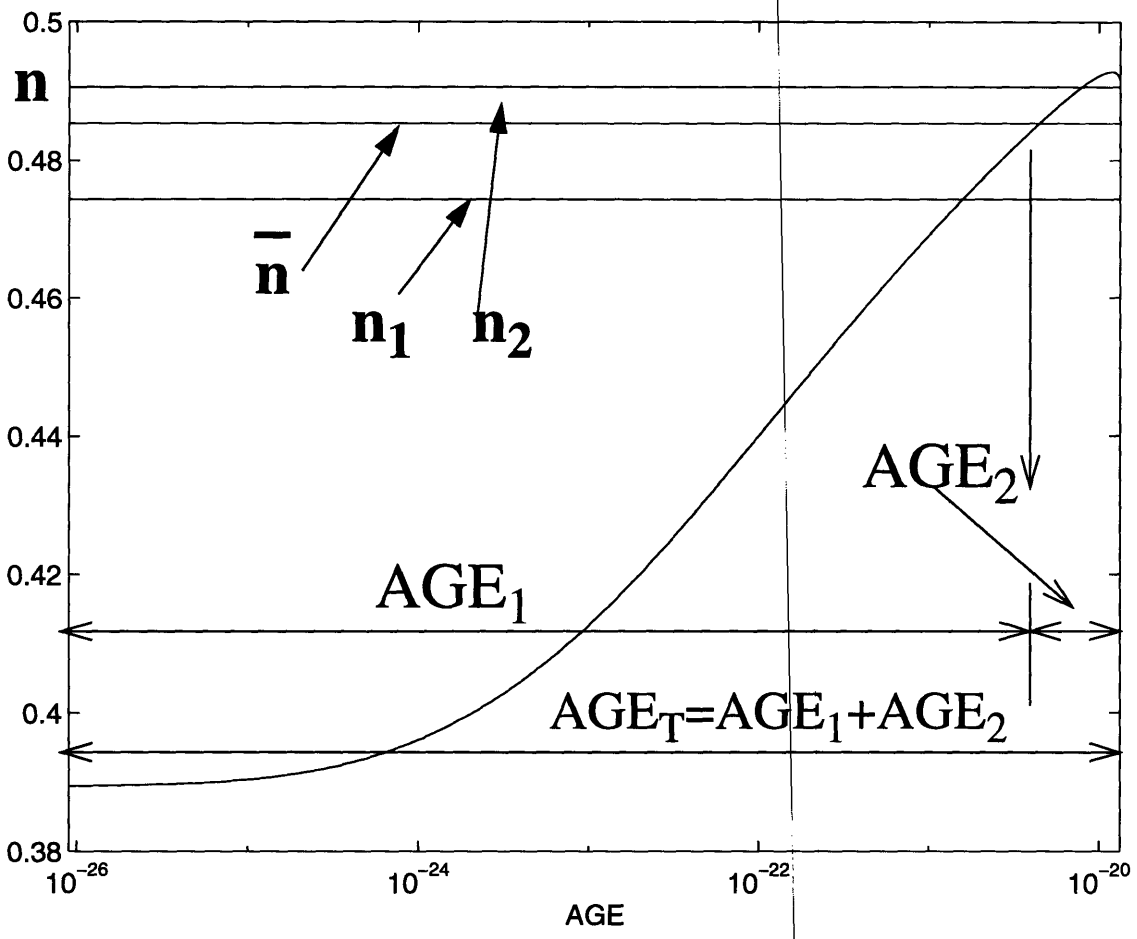


**Figure 4.4:** **n** versus cumulative AGE(semilog plot)

partitioning line $\mathbf{n}=\bar{\mathbf{n}}$ respectively. The equations for $\mathbf{n_1}$ and $\mathbf{n_2}$ are defined below:

$$n_1 = \frac{\sum_{0 < AGE < AGE_1} n_i \cdot (AGE_i)}{AGE_1} \tag{4.6}$$

$$n_2 = \frac{\sum_{AGE_1 < AGE < AGE_T} n_i \cdot (AGE_i)}{AGE_2} \tag{4.7}$$

The equations that describes the two asymptotes generated from the initial asymptote are:

$$\Delta D_1(NT) = \left(AGE_1 \times N\right)^{n_1} \tag{4.8}$$

$$\Delta D_2(NT) = \left(AGE_2 \times N\right)^{n_2} \tag{4.9}$$

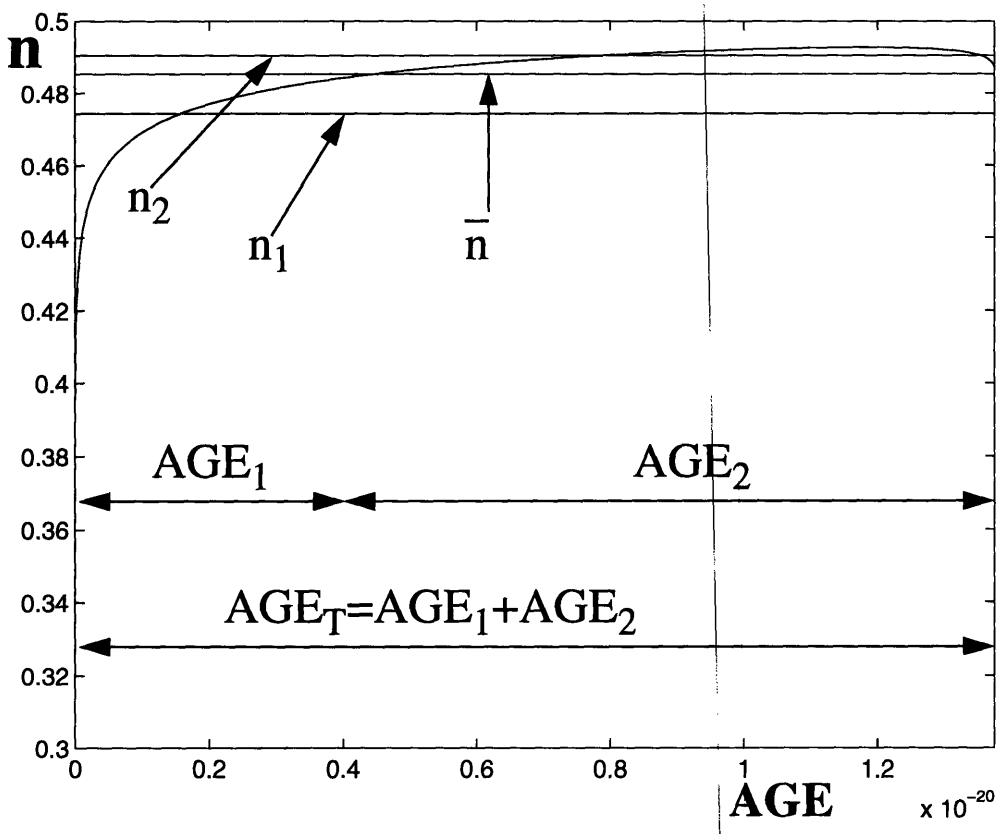


**Figure 4.5: n** versus cumulative AGE(linear scale)

Figure 4.4 is plotted again in Figure 4.5 with a linear scale.

## 4.4 Identifying the Dominant Asymptote

The next step in the algorithm is to identify the dominant asymptote at the time of interest. The dominant asymptote indicates the highest degradation at the time point of interest. Figure 4.6 illustrates the procedure of identifying the dominant asymptote at the time of interest ($N=N_{sp}$). The initial asymptote is partitioned using the method described in the previous section to generate two asymptotes $A_1$ and $A_2$. The 3 asymptotes, the orig-

inal and the two new asymptotes, are compared at the time-point of interest. In Figure 4.6, asymptote $A_1$ dominates at the time of interest.

Next, each of the asymptotes $A_1$ and $A_2$ are then partitioned further using the same procedure for the initial asymptote. This is done to ensure that a maximum degradation has been reached. $A_1$ is partitioned into $A_{11}$ and $A_{12}$. $A_2$ is partitioned into $A_{21}$ and $A_{22}$.

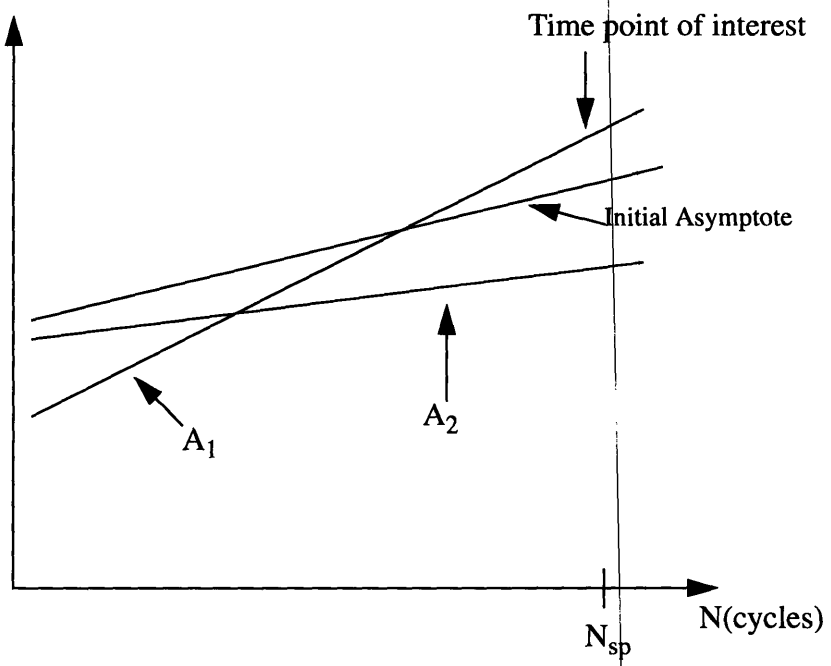


**Figure 4.6:** Identifying the dominant asymptote

After each partitioning, the algorithm compares all the asymptotes and identifies the dominant asymptote at the time of interest. The process of partitioning each of the generated asymptotes and noting the dominant asymptote is repeated until the partitioning results in one of the AGE quantities being very small:

$$AGE_1 \ll AGE_2 \tag{4.10}$$

or:

$$AGE_2 \ll AGE_1 \tag{4.11}$$

The tolerance used for the AGE quantities was if any one of the AGE<$1x10^{-50}$ then the partitioning stopped. When the partitioning of all the asymptotes stopped, then the degradation at the time point of interest is defined by the most dominant asymptote that has been identified. The notation used for the dominant asymptote is $\mathbf{n=n_{dom}}$ and AGE=AGE-$_{dom}$. The degradation at the time point of interest($N=N_{sp}$) is then $\Delta D(N_{sp}T) = (AGE_{dom} \times N)^{n_{dom}}$.

For calculating the lifetime of the device, the same steps used for degradation calculation are followed except that the goal of the algorithm for lifetime calculation is to find the asymptote that will minimize the lifetime(at the given lifetime definition $\Delta D_f$). Thus, for each asymptote, the lifetime will be calculated instead of the degradation using Equation 4.12. The asymptote with the minimum lifetime will determine the lifetime of that device.

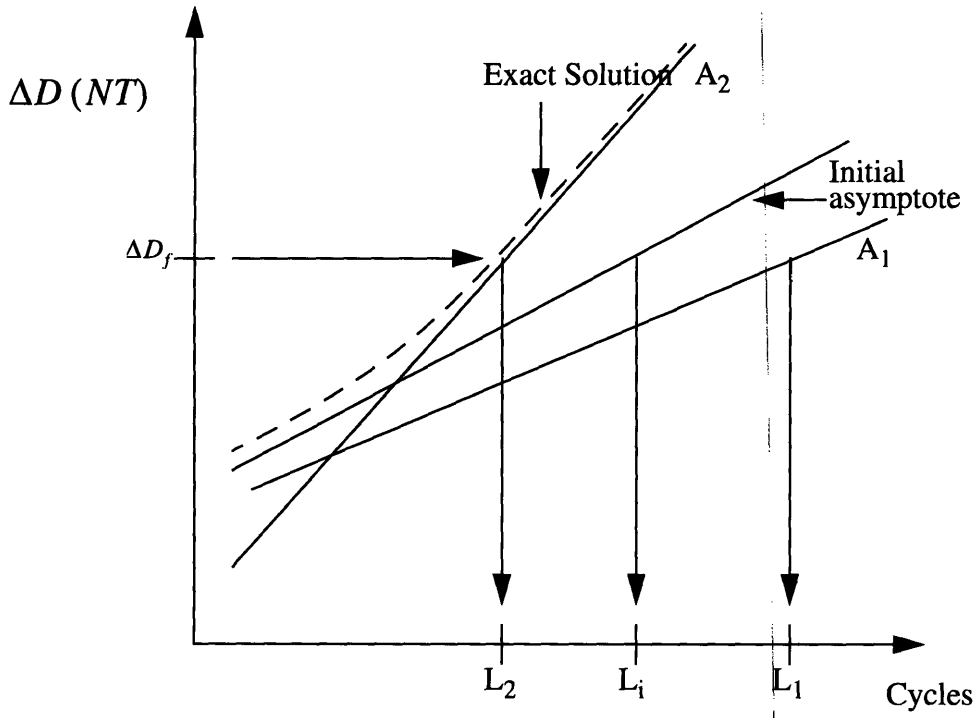


**Figure 4.7:** Lifetime calculation using dominant asymptote

In Figure 4.7 the initial asymptote and two asymptotes($A_1$ and $A_2$) generated by partitioning the initial asymptotes are shown. The lifetime of the device is calculated by finding the asymptote with the minimum lifetime using the same algorithm for the calculation

$$\tau = \frac{(\Delta D_f)^{\frac{1}{n}} \cdot T}{AGE} \tag{4.12}$$

of degradation. From Figure 4.7, asymptote $A_2$ predicts the smallest lifetime($L_2<L_i<L_1$) at the lifetime definition $\Delta D_f$. If the asymptotes in Figure 4.7 represent all the possible asymptotes that can result from partitioning the initial asymptote then the lifetime is defined by asymptote $A_2$. The lifetime defined by asymptote $A_2$ is the closest to the exact lifetime defined by the dashed line which represents the exact degradation. In general, the asymptotes may have to be split many times before the minimum lifetime is found.

## 4.5 Two-step and Inverter Examples

**Two-step**: The two-step example can be used to illustrate why the dominant asymptote method is a more accurate method for predicting the AC degradation of a MOSFET. There are two domains of interest for the two-step example where two different degradation components dominate depending on the time of interest. In Figure 4.8, the simulated degradation for the two step waveform using the stress conditions outlined in Figure 3.1 is shown along with the current $\mathbf{n_{eff}}$ method. In addition, the dominant asymptotes defined using the new method are shown in comparison. The equations that describes the two dominant asymptotes are as follows:

for $N < 10^5 cycles$:

$$\Delta D(NT) = \left(1.45 \times 10^{-23} \times N\right)^{0.337} \tag{4.13}$$

for $N > 10^5 cycles$:

$$\Delta D(NT) = \left(7.2 \times 10^{-18} \times N\right)^{0.495} \tag{4.14}$$

Using the current $\mathbf{n_{eff}}$ method, the equation for the line as shown in Figure 4.8 from Equation 3.18-19:

$$\Delta D(NT) = A_{eff} \cdot (NT)^{n_{eff}} \cong \left(\frac{AGE_1}{T}\right)^{n_1} \cdot (NT)^{n_1} \cong \left(AGE_1 \cdot N\right)^{n_1} \tag{4.15}$$

$$\cong \left(1.45 \times 10^{-23} \times N\right)^{0.337} \tag{4.16}$$

Thus for N<$10^5$ cycles, the current $\mathbf{n_{eff}}$ and the new method predict roughly the same amount of degradation. But for N>$10^5$ cycles, the new method models the AC degradation closer to the exact simulation data as shown in Figure 4.8, while the current $\mathbf{n_{eff}}$ method deviates gradually away from the exact data. This is because the dominant degradation rate **n**

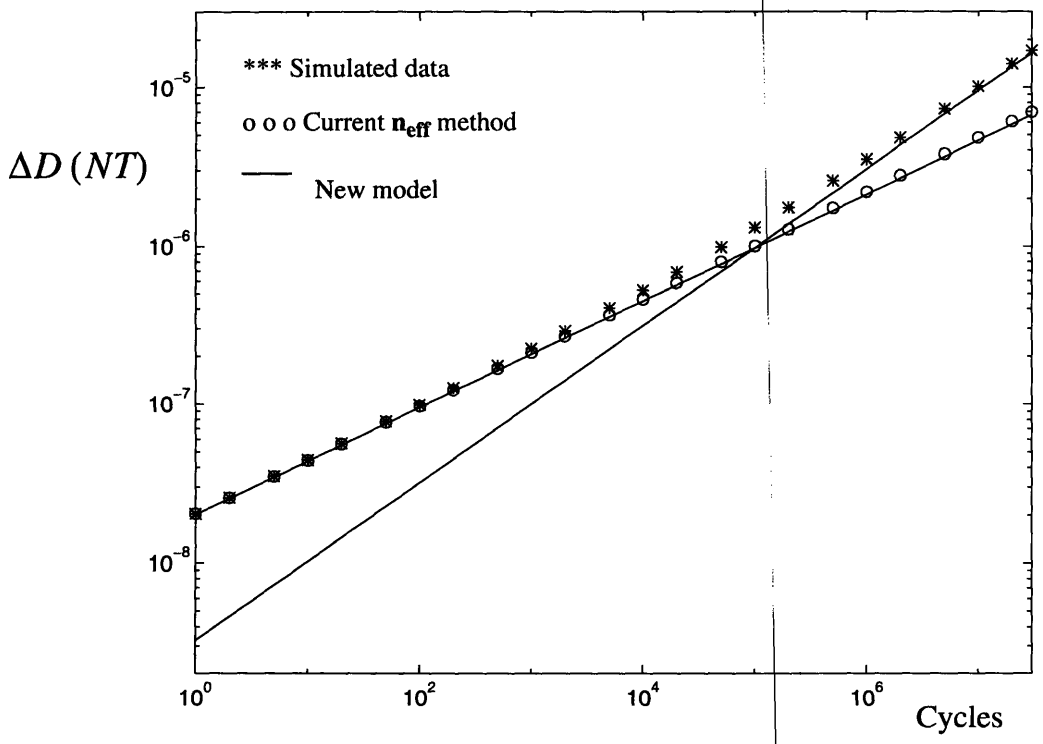


**Figure 4.8:** Two-step simulation along with current $\mathbf{n_{eff}}$ and new model

changes after $N>10^5$ cycles. The degradation rate $\mathbf{n=n_2}$=0.495 dominates for $N>10^5$ cycles and thus Equation 4.14 using the new method predicts the degradation more accurately.

**Inverter:** An inverter example shown in Figure 3.4 is also used to demonstrate the new algorithm. For this example $C_L$=1pF and $\alpha$ = 5 V/ns. In Figure 4.9, the asymptotes generated by the partitioning method are shown for the typical inverter simulation along with the simulated data. The initial asymptote in Figure 4.9 is labeled accordingly and the other asymptotes generated from splitting the initial asymptote and other resulting asymptotes are shown. The degradation of the NMOS in the inverter is evaluated at $10^5$ cycles, where

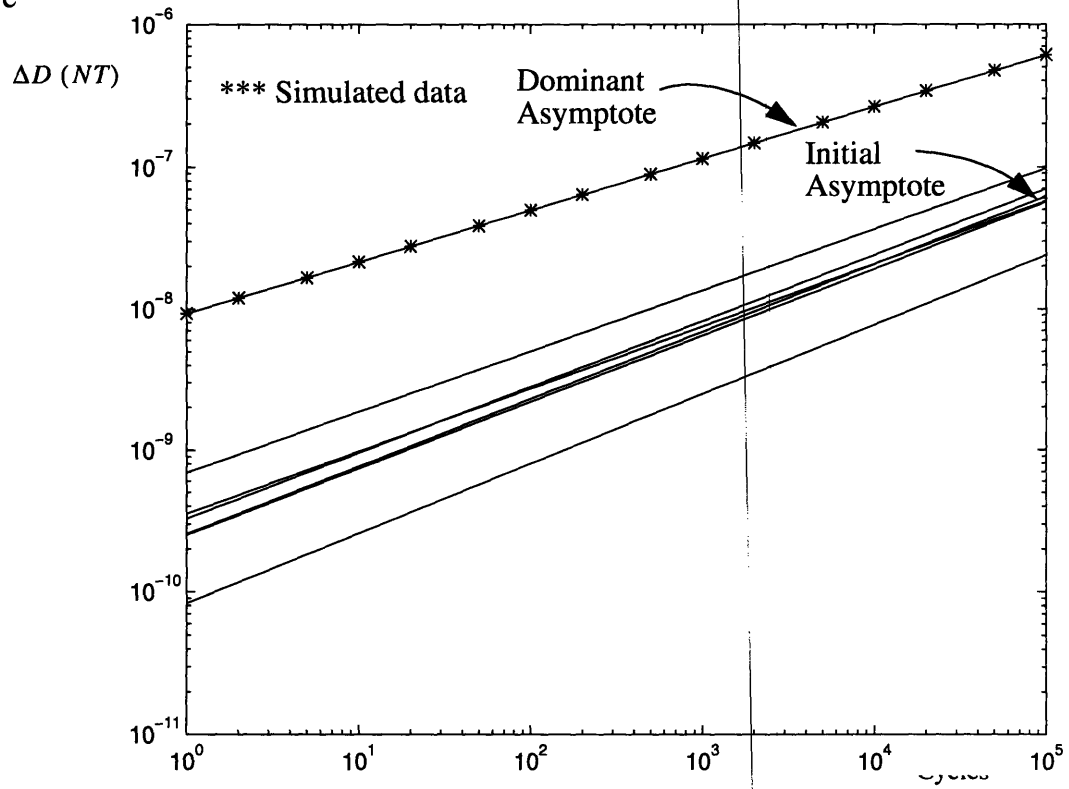


**Figure 4.9:** Demonstration of the algorithm for an inverter

the process of splitting the asymptotes results in the identification of a dominant asymptote. The AC degradation at $10^5$ cycles is determined by this dominant asymptote.

For this example, table 4.1 shows the numerical result of the asymptote splitting and the degradation at $10^5$ cycles. The first column lists the notation used for the asymptotes. The second two columns of the table represents the AGE and **n** respectively that define each of the asymptotes shown in Figure 4.9. The last column on the table is the degradation value predicted by the asymptote at $10^5$ cycles. The degradation values in table 4.1 fluctuate with each new asymptotes and eventually one of the asymptotes predicts the maximum degradation. The maximum degradation is shown with an asterisk.

**Table 4.1: Numerical results for NMOS in inverter at N=$10^5$ cycles**

| Asymptote Notation | AGE | n | $\Delta D\,(NT) = (AGE \times N)^n$ |
|---|---|---|---|
| $A_o$, $n_o$ | 7.7680e-21 | 4.7702e-01 | 6.199766e-08 |
| $A_1$, $n_1$ | 3.2480e-21 | 4.9202e-01 | 2.395550e-08 |
| $A_2$, $n_2$ | 4.5199e-21 | 4.6623e-01 | 7.010689e-08 |
| $A_{21}$, $n_{21}$ | 4.0187e-21 | 4.7075e-01 | 5.654227e-08 |
| $A_{22}$, $n_{22}$ | 5.0122e-22 | 4.3003e-01 | 9.784152e-08 |
| $A_{221}$, $n_{221}$ | 4.2396e-22 | 4.4220e-01 | 5.75444e-08 |
| $A_{222}$, $n_{222}$ | 7.7264e-23 | 3.6324e-01 | 6.084555e-07*** |

The first line in table 4.1 represents the initial asymptote. For this example, the initial maximum degradation is ~ 6.2e-8. After the first split, the maximum degradation is changed to ~7e-8. The maximum degradation increases to ~9.8e-8 in the second split and to ~6.1e-7 in the last split. The asymptote labeled "dominant asymptote" represents the maximum degradation after the three splits. The degradation at $10^5$ cycles is ~6.1e-7 according to the new algorithm. This result agrees well with the simulated degradation at $10^5$ cycles which is approximately 6.11e-7.

## 4.6 Implementation of the New Model using a C program

This section describes the detail of how the algorithm demonstrated in the previous section can be implemented using the C programming language. The program that implements the algorithm is essentially a post-processor, which means that it executes the code using results from another program. The program code is listed in appendix A. In this case the results that the program needs is the output from SPICE and the output from BERT(BErkeley Reliability Tools). In addition it requires the model parameters for **m,n,H** degradation parameters and other information relevant to the circuit under test.

The first input to the program is the output of SPICE. This output from SPICE has the data for the voltage waveform($V_g$ , $V_d$), and drain current $I_d$. The SPICE input netlist contains commands to print out the $V_g$, $V_d$ and $I_d$ value at each timestep for each NMOS device in the circuit. The voltage waveforms ($V_g$, $V_d$) are needed to calculate the $V_{gd}$ (hence $E_{ox}$) bias dependencies of the degradation parameters **m,n,H** as described in chapter 2. The drain current $I_d$ is used to calculate the variable AGE defined in Equation 2.8. The drain current in SPICE is calculated using the BSIM level 13 SPICE model.

The other inputs to the program is the substrate current output from BERT. BERT is used to calculate the substrate current using the substrate current model described in chapter 2. BERT calculates the substrate current at the timestep determined by the SPICE simulation[5]. The substrate current data is used to calculate the AGE variable.

Other inputs to the program include the SPICE input netlist file, which has information pertaining to the circuit, and other user inputs. These information include the device width W, which is necessary to calculate AGE and the period T which is necessary for calculating the lifetime of the devices. The user input includes the specification of the lifetime definition, user specified time point(for calculation of degradation), and the model parameters for **m,n,H** and its bias dependencies.

The role of the program is to take all these inputs and process it to produce the output which is the degradation and lifetime of each devices at the user specified time point and lifetime definition respectively. The overall structure of the program is shown in Figure 4.10 with the inputs and output for the program.

In stage 1 of Figure 4.10, the output of SPICE and BERT for the NMOS devices are processed so that they can be readable by the program. The device width, W, and the period of the waveform T can be obtained from the **SPICE** netlist which is another input.

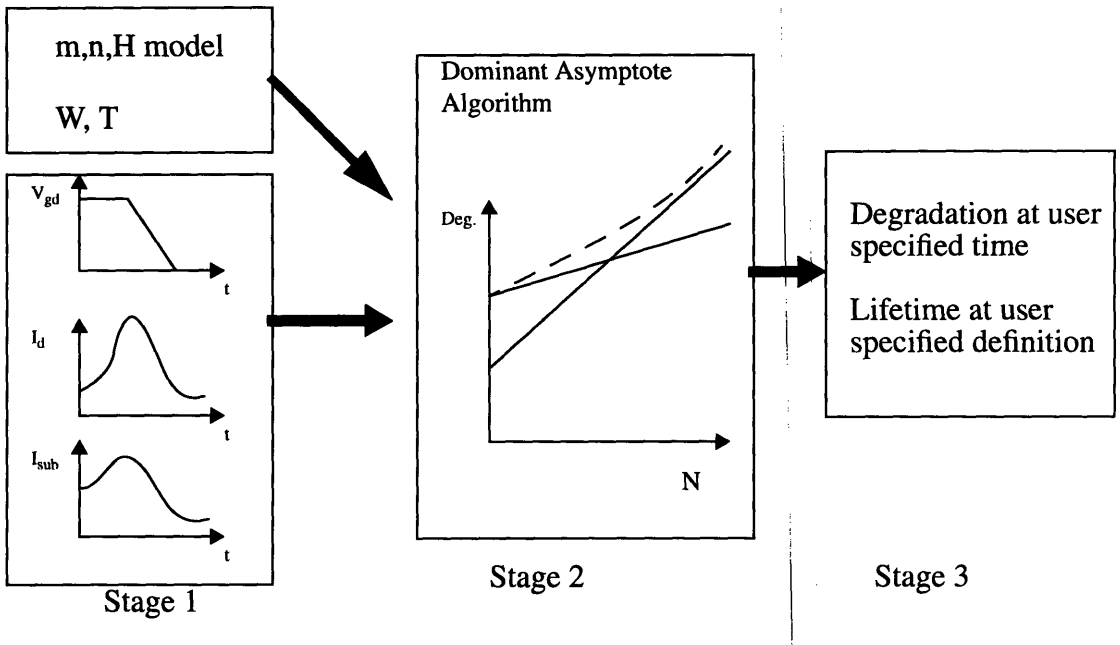


**Figure 4.10:** Structure of the new model

The degradation parameters along with their bias dependency are also the inputs to the second stage of the program. These inputs suffice to do the calculations for AGE and **n** which are necessary for formulating the degradation asymptotes.

In stage 2, the processed information can be used to implement the new model. At each timestep of the SPICE waveform, the program calculates $V_{gd}=V_g-V_d$ in order to cal-

culate $E_{ox}$. Then it calculates the degradation parameters **n,m,H** as a function of the bias $V_{gd}$ at this timestep. The next step is to calculate the variable AGE=$f$($I_d$, $I_{sub}$, W,m,H). To calculate AGE, which is the integral of the waveform shown in Figure 2.2, the substrate current $I_{sub}$ and drain current $I_d$ are read from the output of the SPICE and BERT simulation at each timestep. The AGE in each timestep is calculated by using the trapezoidal method. The process of computing **n** and AGE is done for each timestep in the SPICE analysis and the result is stored in data files.

These data are then used to calculate the initial asymptote. The partitioning of the initial asymptote is accomplished by a function that takes arrays of data with all the **n**'s and AGE at each timestep and creates two sets of arrays by partitioning the **n**'s and the AGE with $\mathbf{n}=\bar{\mathbf{n}}$ . These arrays can be used to determine the two asymptotes generated from the initial asymptote. The partitioning is continued until the asymptotes cannot be partitioned further. The maximum degradation and the lifetime are eventually determined and stored in a file specified by the user in stage 3. For multiple devices in a circuit, the same procedure is repeated and the results for each transistor are stored in one user specified file.

# Chapter 5

# Analysis of the New Model

## 5.1 Asymptotic Behavior of the Exact Degradation

The asymptotic behavior of the exact AC degradation equation can be analyzed for the two-step case. A proof can be shown to illustrate that the exact solution tends to track one of the two asymptotes for the two-step wave form.

The set of conditions necessary for this proof are outlined below:

$$AGE_1^{\frac{n_1}{n_2}} \gg AGE_2 \tag{5.1}$$

$$n_1 < n_2 \tag{5.2}$$

Equation 3.8 is revisited for this analysis. In Equation 3.8, there are two components that are competing, namely the terms involving powers of $\frac{n_1}{n_2}$ and powers of $\frac{n_2}{n_1}$. The terms involving powers $\frac{n_1}{n_2}$ can be defined as $x$ while the terms involving powers of $\frac{n_2}{n_1}$ can be defined as $y$. Since $n_1<n_2$, the ratio of $\frac{n_2}{n_1}>1$ and $\frac{n_1}{n_2}<1$.

In order to determine the dominant component, the term $x^{\frac{n_1}{n_2}}$ has to be compared to $AGE_2$ and the term $y^{\frac{n_2}{n_1}}$ has to be compared to $AGE_1$. Initially, the terms $x$ and $y$ are very small because the degradation is small. Since y is raised to a power greater than one, the term $y^{\frac{n_2}{n_1}}<<1$ and from Equation 5.1 the terms $x^{\frac{n_1}{n_2}}>>AGE_2$. Using these approximations, Equation 3.8 can be simplified for $N<N_{int}$:

$$\Delta D(NT) = \left( \left( [\Delta D((N-1)T)]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \right)^{n_2} \tag{5.3}$$

$$\cong \left( \left( \left[ \Delta D\left( (N-1)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} \right)^{n_2} = \left( \left[ \Delta D\left( (N-1)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{n_1} \tag{5.4}$$

$$\Delta D\left( (N-1)\,T \right) = \left( \left( \left[ \Delta D\left( (N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \right)^{n_2} \tag{5.5}$$

$$\cong \left( \left( \left[ \Delta D\left( (N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} \right)^{n_2} \cong \left( \left[ \Delta D\left( (N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{n_1} \tag{5.6}$$

Plugging Equation 5.6 into Equation 5.3:

$$\Delta D\,(NT) \cong \left( \left[ \left( \left[ \Delta D\left( (N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{n_1} \right]^{\frac{1}{n_1}} + AGE_1 \right)^{n_1} \tag{5.7}$$

$$\cong \left( \left[ \Delta D\left( (N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 + AGE_1 \right)^{n_1} \tag{5.8}$$

By the same argument the degradation at t=(N-2)T can be found:

$$\Delta D\left( (N-2)\,T \right) = \left( \left( \left[ \Delta D\left( (N-3)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \right)^{n_2} \tag{5.9}$$

$$\cong \left( \left( \left[ \Delta D\left( (N-3)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} \right)^{n_2} = \left( \left[ \Delta D\left( (N-3)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{n_1} \tag{5.10}$$

where Equation 5.10 is derived using Equation 5.1. Plugging Equation 5.10 into 5.8:

$$\Delta D\,(NT) \cong \left( \left( \left( \left[ \Delta D\left( (N-3)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{n_1} \right)^{\frac{1}{n_1}} + 2AGE_1 \right)^{n_1} \tag{5.11}$$

$$= \left( \left[ \Delta D\left( (N-3)\,T \right) \right]^{\frac{1}{n_1}} + 3AGE_1 \right)^{n_1} \tag{5.12}$$

Thus Equation 5.12 can be iterated *k* times to show that the degradation initially tracks the

asymptote with rate $\mathbf{n_1}$:

$$\Delta D(NT) \cong \left( \left[ \Delta D\left( (N-k)\,T \right) \right]^{\frac{1}{n_1}} + kAGE_1 \right)^{n_1} \tag{5.13}$$

if $k \to N$ then Equation 5.13 becomes:

$$\Delta D(NT) \cong \left( \left[ \Delta D(0) \right]^{\frac{1}{n_1}} + NAGE_1 \right)^{n_1} = \left( N \times AGE_1 \right)^{n_1} \tag{5.14}$$

Thus initially the degradation tracks the asymptote with rate $\mathbf{n_1}$.

For N large, the terms $y^{\frac{n_2}{n_1}}$ starts to dominate because $y^{\frac{n_2}{n_1}} >> \mathrm{AGE}_1$ and $\mathbf{n_2 > n_1}$. For this case, the degradation at t=NT can be defined:

$$\Delta D(NT) = \left( \left( \left( \left( \left[ \Delta D(N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \right)^{\frac{n_2}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \Bigg)^{n_2} \tag{5.15}$$

$$\Delta D(NT) \cong \left( \left( \left( \left( \left[ \Delta D(N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \right)^{\frac{n_2}{n_1}} \right)^{\frac{n_1}{n_2}} + AGE_2 \Bigg)^{n_2} \tag{5.16}$$

$$= \left( \left( \left[ \Delta D(N-2)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 + AGE_2 \Bigg)^{n_2} \tag{5.17}$$

$$= \left( \left( \left[ \Delta D\left( (N-2)\,T \right) \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + 2 \cdot AGE_2 \Bigg)^{n_2} \tag{5.18}$$

For the second iteration:

$$\Delta D\left( (N-2)\,T \right) = \left( \left( \left( \left( \left[ \Delta D\left( (N-4)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \right)^{\frac{n_2}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \Bigg)^{n_2} \tag{5.19}$$

$$\cong \left( \left( \left( \left( \left[ \Delta D\left( (N-4)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 \right)^{\frac{n_2}{n_1}} \right)^{\frac{n_1}{n_2}} + AGE_2 \Bigg)^{n_2} \tag{5.20}$$

$$= \left( \left( \left[ \Delta D\left( (N-4)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + AGE_2 + AGE_2 \right)^{n_2} \tag{5.21}$$

$$= \left( \left( \left[ \Delta D\left( (N-4)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + 2 \cdot AGE_2 \right)^{n_2} \tag{5.22}$$

Plugging Equation 5.22 into Equation 5.18:

$$\Delta D(NT) \cong \left( \left( \left( \left( \left[ \Delta D\left( (N-4)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + 2 \cdot AGE_2 \right)^{\frac{n_2}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + 2 \cdot AGE_2 \right)^{n_2} \tag{5.23}$$

$$= \left( \left( \left( \left( \left[ \Delta D\left( (N-4)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + 2 \cdot AGE_2 \right)^{\frac{n_2}{n_1}} \right)^{\frac{n_1}{n_2}} + 2 \cdot AGE_2 \right)^{n_2} \tag{5.24}$$

$$= \left( \left( \left[ \Delta D\left( (N-4)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + 4 \cdot AGE_2 \right)^{n_2} \tag{5.25}$$

Iterating $k$ times the degradation at t=NT is:

$$\Delta D(NT) \cong \left( \left( \left[ \Delta D\left( (N-2k)\,T \right) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + 2k \cdot AGE_2 \right)^{n_2} \tag{5.26}$$

If $k \to \frac{N}{2}$ the degradation at t=NT for N large is;

$$\Delta D(NT) \cong \left( \left( \left[ \Delta D(0) \right]^{\frac{1}{n_1}} + AGE_1 \right)^{\frac{n_1}{n_2}} + N \cdot AGE_2 \right)^{n_2} \tag{5.27}$$

$$\Delta D(NT) \cong \left( AGE_1^{\frac{n_1}{n_2}} + N \cdot AGE_2 \right)^{n_2} \cong \left( N \cdot AGE_2 \right)^{n_2} \tag{5.28}$$

Thus for N large Equation 5.28 states that the exact degradation tracks the asymptotes

with degradation rate $\mathbf{n=n_2}$.

## 5.2 Validity of the Approximation

For the two-step case, the accuracy of the analysis can be evaluated by using actual simulated data. The basis for the approximations used in the analysis can be redefined in terms of two ratios $r_1$ and $r_2$ where ratio $r_1$ is defined as follows:

$$r_1 = \frac{y^{\frac{n_2}{n_1}}}{AGE_1} \tag{5.29}$$

and ratio $r_2$ is defined as follows:

$$r_2 = \frac{x^{\frac{n_1}{n_2}}}{AGE_2} \tag{5.30}$$

The approximation in the analysis depended on which of the ratios were bigger. For $N<N_{int}$, the term $x^{\frac{n_1}{n_2}} >> AGE_2$ and $y^{\frac{n_2}{n_1}} << 1$. Thus initially the ratio $r_2 >> r_1$. For large N, $y^{\frac{n_2}{n_1}} >> AGE_1$ and $x^{\frac{n_1}{n_2}} << AGE_2$. In this case, the ratio $r_1 >> r_2$. These two ratios can then be used to determine which of the two asymptotes dominate, hence which asymptote the true solution will track. If $r_1 >> r_2$, i.e. N large, then the exact solution will track the asymptote with rate $\mathbf{n_2}$. If $r_2 >> r_1$, for N small, then the exact solution will track the asymptote with rate $\mathbf{n_1}$.

For the two-step example the plot of the two ratios using the simulated data from Figure 3.3 are shown in Figure 5.1. From Figure 5.1, for $N<10^5$ cycles $r_2 >> r_1$ which implies that the dominant rate is $\mathbf{n_1}$. For $N>10^5$ cycles $r_2 << r_1$ and the dominant rate is $\mathbf{n_2}$ as expected for this particular two-step example. Thus the approximation used in the analysis is valid because it agrees with figure 5.1 .

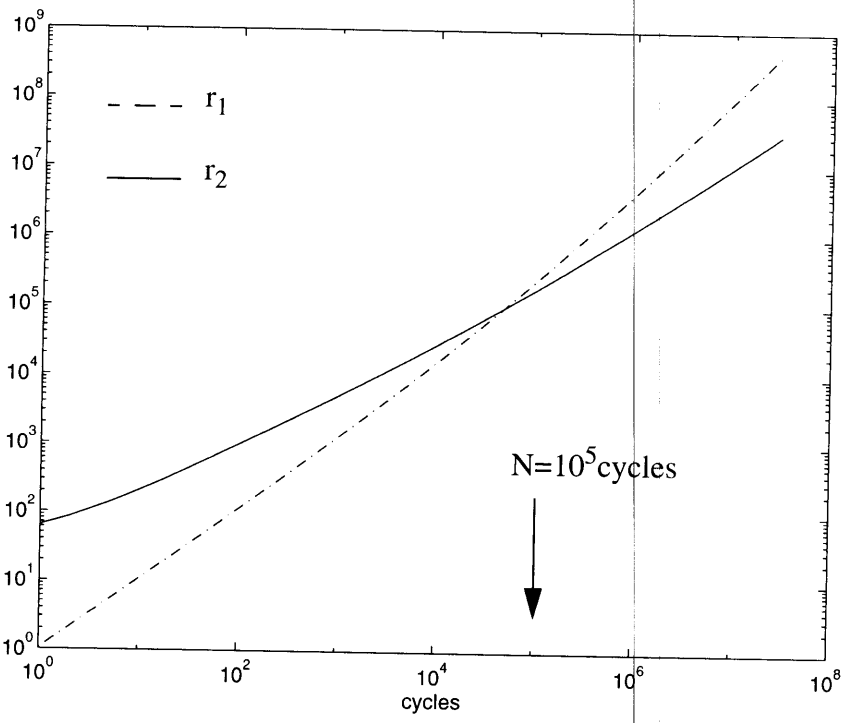


**Figure 5.1:** Ratios $r_1$ and $r_2$ for the two-step example

# Chapter 6

# Simulation Results and Discussion

## 6.1 Simulation result for NMOS in Inverter

To test the new algorithm, a long simulation was performed - examining the AC degradation of the NMOS transistor of an inverter under certain input waveform conditions and capacitative loads. For this test case, the inverter described in Figure 3.4 was used. The simulation for the inverter was done for several cycles(up to $2x10^8$ cycles with period T=10ns, $\frac{W}{C_L}$=400 $\frac{\mu m}{pF}$, $\alpha = 10\frac{V}{ns}$) to show how the new method better predicts the AC hot-electron degradation in comparison to the current $\mathbf{n_{eff}}$ method. Figure 6.1 shows the same AC simulation points shown in Figure 3.6 calculated using the exact iterative AC degradation equations. The degradation predicted by the new and current $\mathbf{n_{eff}}$ method is also shown in Figure 6.1.

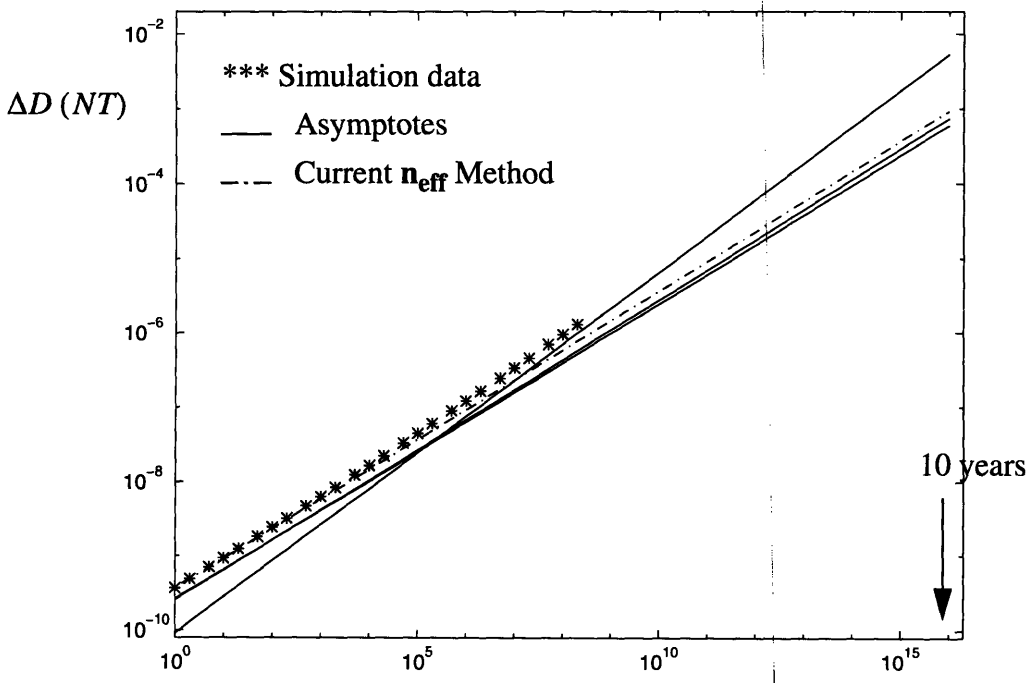


**Figure 6.1:** Simulation result for NMOS in inverter
($\frac{W}{C_L}$ = 400$\frac{\mu m}{pF}$, $\alpha = 10\frac{V}{ns}$)

Initially the current $\mathbf{n_{eff}}$ method predicts an accurate degradation value but after a large number of cycles (N> $10^6$ cycles) the current $\mathbf{n_{eff}}$ method starts to diverge away from the exact solution while the method based on the dominant asymptote stays close to the simulated data.

A similar simulation was done with an inverter for different values of the design parameters $\alpha$ and $\frac{W}{C_L}$. The values for the next simulation is $\alpha = 1\frac{V}{ns}$ and $\frac{W}{C_L} = 10\frac{\mu m}{pF}$. The result is shown in Figure 6.2 with the same notations for the data points as Figure 6.1. The main difference between Figure 6.2 and Figure 6.1 is the intersection point of the dominant asymptotes. For Figure 6.1 the intersection point is $N \cong 10^6$ cycles but for the simulation in Figure 6.2 the intersection point occurs much earlier($N \cong 2 \cdot 10^2$ cycles). By making the intersection point earlier, the asymptotic behavior of the simulated data can be more clearly observed. Note, that the dominant asymptote tracks the simulated data accurately beyond the intersection point. Figure 6.3 shows the same data with N extended to

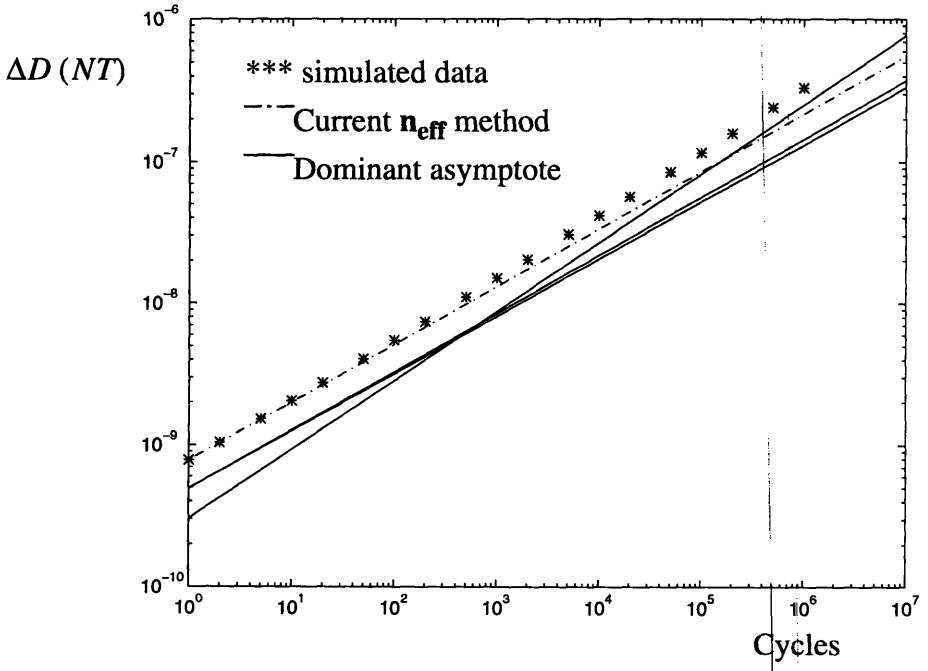


**Figure 6.2:** Simulation result for NMOS in inverter
$(\alpha = 1\frac{V}{ns}, \frac{W}{C_L} = 10\frac{\mu m}{pF})$

$10^5$ cycles to show the overestimation of lifetime using the current $\mathbf{n_{eff}}$ method.

The overestimation in AC lifetime in the current $\mathbf{n_{eff}}$ method compared with the new method was calculated for inverters over a wide range of design parameters. The design variables $W/C_L$ and the ramp rate $\alpha$ were varied for the inverter shown in Figure 3.4 over a wide range. The lifetime definition for the simulation was at $\Delta\left(\frac{I_d}{I_{do}}\right) = 3\,\%$. The range on the design parameters are $1 < \alpha < 10$ (V/ns) and $40 < \frac{W}{C_L} < 400$ ($\mu$m/pF). The 3-d plot of these data is shown in Figure 6.4 with the x and y axis being the design parameters(W/ $C_L$ and $\alpha$ (ramp rate)). The z-axis is the ratio between lifetime ($\left.\frac{\tau_{old}}{\tau_{new}}\right|_{\Delta\left(\frac{I_d}{I_{do}}\right)=3}$)predicted by the current $\mathbf{n_{eff}}$ method($\tau_{old}$) and lifetime predicted by the new method. The purpose of this plot is to show that the current $\mathbf{n_{eff}}$ method of predicting lifetime based on using only the first two waveform cycles can drastically overestimate the lifetime of the NMOS device over a wide range of design parameters. For this example, the lifetime can be over-estimated by as much as two orders of magnitude, as seen in Figure 6.4.

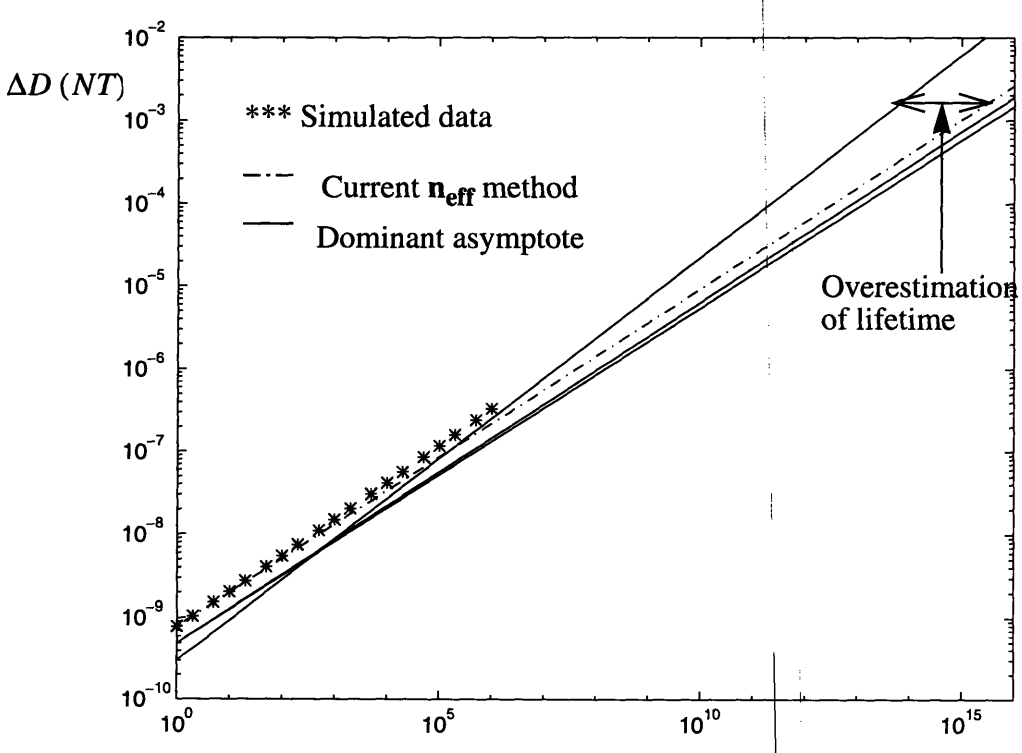


**Figure 6.3:** Overestimation of lifetime using the current $\mathbf{n_{eff}}$ method

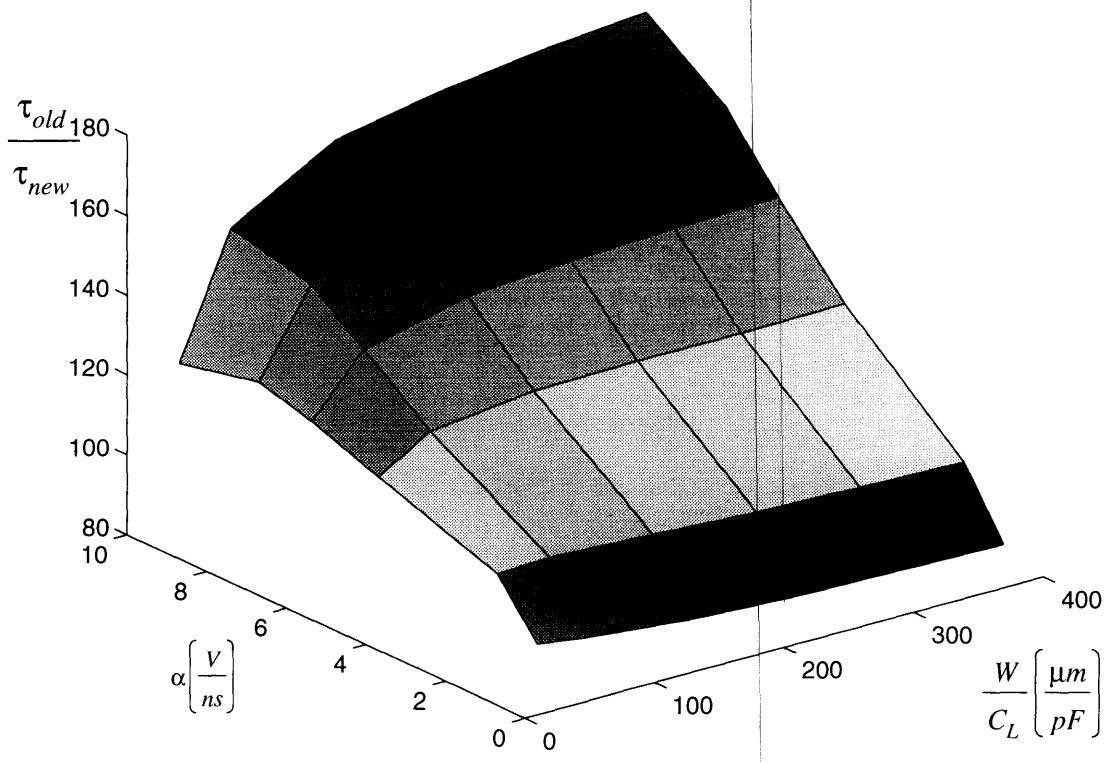


**Figure 6.4:** Ratio of AC lifetime predicted by the current $\mathbf{n_{eff}}$ and new method based on dominant asymptotes.

# Chapter 7

# Conclusion

This thesis focused on analyzing AC lifetime prediction models for NMOSFETs. The quasi-static model has been used to predict AC degradation due to hot-carriers. It has been shown that predicting AC degradation with the quasi-static model and its associated iterative degradation equations is infeasible because of limitations on the allowable computation time. The current model for AC hot-carrier degradation predicts AC degradation and lifetime by extrapolating from the first two cycles in the AC waveform to the specified time-point. Significant extrapolation error can be introduced, however, if the specified time-point is way beyond the first two cycles, the degradation rate for the first few cycles may not necessarily be the degradation rate after many cycles.

The new model for AC hot-carrier degradation utilizes the concept of the "dominant asymptote". The new model predicts AC degradation by identifying the dominant asymptote at the time-point of interest. It has been shown that AC degradation exhibits an asymptotic behavior and that the degradation asymptote accurately models the variance in the degradation rate over time. Because the new model adjusts to the dynamic behavior of AC degradation, the amount of error in predicting AC degradation is reduced.

Simulation data for both the two-step and the inverter examples have shown that the new model predicts AC degradation more accurately than the current method. For the two-step example, the current method overestimates the lifetime by as much as two orders of magnitude in comparison to the lifetime predicted by the new model. The new model can also accurately predict the lifetime of NMOSFETs in inverters for a wide range of circuit-design space. Thus, the new model, based on the dominant asymptote, is suggested as a better way of predicting AC hot-electron degradation/lifetime.

# Appendix A

# C Program for the New Algorithm

```
#include <stdio.h>
#include <math.h>
#include <string.h>
struct agedata{  /* agedat is a structure that stores the result of splitting an asymptote */
  double n1;    /* age1,n1,age2,n2 is the resultant components of the "twins" */
  double n2;    /* ct is a counter used as an index */
  double age1;  /* ints is an array that stores the intersection points of the n vs. AGE plot */
  double age2;  /* where the intersection points are determined by the n average */
  int ct;       /* f is a flag used for the partioning  */
  int f;
  int ints[100];
};

#define MAXT 16  /* MAXT defines the maximum number of transistors the program can accomodate */
#define TOL 1e-50 /* TOL is a tolerance on how small AGE can be */
double totage(int, int,int,double []);
int compare(char [],char [][10],int);
char * fwidth(char [],FILE *);
struct agedata partition(double [],double [],double,int);
double integral(int,int,double [],double []);
double findarea(int , int, int,double [],double []);
void cumage(double,double [],double [],double [],double [],int,double [],char [],double []);
void degcalc(int,double ldef,double cyc,double year,double p,double ngdo,double ngd1,double ngd2,double mgdo,dou-
ble mgd1, double mgd2,double hgdo,double hgd1,double hgd2,double tox,double vfb,int);
int newdeg(char [],double,double,int,double[],double[],double *, double *, double *,FILE *);
void olddeg(double [],double [],int,FILE *,FILE *,double,double,double,char * , double, double);
double finddeg(double [],double [], int, int);
int filter(int *);
double lifetime(double [],double [],double,int,double *,double *,double *);
main()
{
  FILE * NEW,* OLD, * NOUT,* BOUT;
  char new[30],bert[30];
  int trannum,numstep;
  char * width[16];
  double p,ngdo,ngd1,ngd2,mgdo,mgd1,mgd2,tox,vfb,* w;
  double hgdo,hgd1,hgd2;
  int i,num,j;
  double ldef;
  double agetot,year,cyc;
/* enter the model parameters for m,n,H along with the oxide */
  /* dependencies ngd, hgd, ..... */
/*  printf("enter ngdo ");
  scanf("%lf",&ngdo);
  printf("enter ngd1 ");
  scanf("%lf",&ngd1);
  printf("enter ngd2 ");
  scanf("%lf",&ngd2);
  printf("enter mgdo ");
  scanf("%lf",&mgdo);
  printf("enter mgd1 ");
  scanf("%lf",&mgd1);
```

```
 printf("enter mgd2 ");
 scanf("%lf",&mgd2);
 printf("enter hgdo ");
 scanf("%lf",&hgdo);
 printf("enter hgd1 ");
 scanf("%lf",&hgd1);
 printf("enter hgd2 ");
 scanf("%lf",&hgd2);
 printf("enter tox ");
 scanf("%lf",&tox);
 printf("enter vfb (v) ");
 scanf("%lf",&vfb);*/
 /* Abraham's m,n,H model parameters */
 ngdo=0.4344;
 ngd1=-0.0747039;
 ngd2=-0.0239151;
 mgdo=3.703;
 mgd1=1.144;
 mgd2=0.753;
 hgdo=4.0;
 hgd1=0;
 hgd2=0;
 tox=1.35;
 vfb=0.25;

 /* Wenjie's m,n,H model parameters */
 /* ngdo=0.2303;
  ngd1=-0.0345;
  ngd2=-0.0044;
  mgdo=3.376;
  mgd1=2.233;
  mgd2=0.961;
  hgdo=4.3461;
  hgd1=-2.7427;
  hgd2=-0.998;
  tox=1;
  vfb=0;*/
 printf("enter the period (ns) ");
 scanf("%lf",&p);
 printf("enter the year to calculate degradation  : ");
 scanf("%lf",&year);
 printf("enter the lifetime definition (dID/Ido)(%%) : ");
 scanf("%lf",&ldef);
 numstep=filter(&trannum); /* function filter reads the SPICE and BERT input and processes it */
                  /* so that the program can use it to calculate AGE and m,n,H */
                  /* trannum is the number of transistors in the circuit     */
                  /* numstep is the number of timesteps in the SPICE analysis */
 cyc=year*((3600*24*365)/(p*1e-9));
 degcalc(trannum,ldef,cyc,year,p,ngdo,ngd1,ngd2,mgdo,mgd1,mgd2,hgdo,hgd1,hgd2,tox,vfb,numstep);
}

/* degcalc function reads in the waveform(vg,vd,Id,Isub) from the previous files and
   calculates m,n,H=f(vgd) and also AGE at each SPICE timestep by the invocation of the
   function cumage. the function also calculates the lifetime and degradation of each
   device by calling the function newdeg. Results are stored in the file NOUT, BOUT, and OLD*/

void degcalc(int count,double ldef,double cyc,double year,double p,double ngdo,double ngd1,double ngd2,double
mgdo,double mgd1,double mgd2,double hgdo,double hgd1,double hgd2,double tox,double vfb,int numstep)
{
```

```
 double ageT,agetot,mean,deg0,tau;
 double width[16],*cumlage,maxage,maxn,maxdeg=0,minage,minn,minlif=1.00e99;
 char tnum[30],im[2];
 char *
name1[16]={"f1.dat","f2.dat","f3.dat","f4.dat","f5.dat","f6.dat","f7.dat","f8.dat","f9.dat","f10.dat","f11.dat","f12.dat"
,"f13.dat","f14.dat","f15.dat","f16.dat"};
 char *
name2[16]={"ib1.dat","ib2.dat","ib3.dat","ib4.dat","ib5.dat","ib6.dat","ib7.dat","ib8.dat","ib9.dat","ib10.dat","ib11.d
at","ib12.dat","ib13.dat","ib14.dat","ib15.dat","ib16.dat"};
 FILE * f1,* f2,* f3,* f4,* f5,* f6,* f7,* f8,* f9,* f10,* f11,* f12;
 FILE * wfile,*OLD,*NOUT,*BOUT,*cage,*asym;
 double *vgd,*age,*eox,w,* n,* vg,* vd,* t, * id,* ib,* m,* h;
 int i,pts,c,j,z;
 char new[30],bert[30];
 printf("enter the file name to store result(new method): ");
 scanf("%s",new);
 printf("enter the file name to store result(Bert method): ");
 scanf("%s",bert);
 asym=fopen("asymptote.dat","w");
 OLD=fopen(bert,"w");
 strcat(new,".out");
 strcat(bert,".out");
 NOUT=fopen(new,"w");
 BOUT=fopen(bert,"w");
 t=(double *)malloc(numstep*sizeof(double));   /* allocate memory for the all the variables */
 vd=(double *)malloc(numstep*sizeof(double));
 vg=(double *)malloc(numstep*sizeof(double));
 vgd=(double *)malloc(numstep*sizeof(double));
 eox=(double *)malloc(numstep*sizeof(double));
 id=(double *)malloc(numstep*sizeof(double));
 ib=(double *)malloc(numstep*sizeof(double));
 n=(double *)malloc(numstep*sizeof(double));
 m=(double *)malloc(numstep*sizeof(double));
 h=(double *)malloc(numstep*sizeof(double));
 fprintf(NOUT,"%s %e %s  %s  %e %s \n","Lifetime at ",ldef,"%","Deg. at ",year," yrs.");
 fprintf(NOUT,"%s %s  %s  %s\n","Transistor","Lifetime(yr)","Degrdation(%)","Age");
 fprintf(BOUT,"%s %e %s  %s  %e %s \n","Lifetime at ",ldef,"%","Deg. at ",year," yrs.");
 fprintf(BOUT,"%s %s  %s  %s\n","Transistor","Lifetime(yr)","Degrdation(%)","Age");
 wfile=fopen("width.dat","r");
 for (i=0; i<count; i++)          /* read in the width of each device */
  {
   fscanf(wfile,"%lf",&width[i]);
  }                          /* select the numerical integration method */
 printf("what integration method would you like: rectangular(r) trapezoid(t):  ");
 scanf("%s",im);
 for(i=0;i<count;i++)            /* count is the number of transistors  */
  {                        /* for each transistors the degradation and lifetime is calculated */
   f1=fopen(name1[i],"r");
   f2=fopen(name2[i],"r");
   f3=fopen("t.dat","w");
   f4=fopen("n.dat","w");
   f5=fopen("nt.dat","w");
   fscanf(f1,"%s",tnum);
   j=0;
   while(fscanf(f1,"%lf",&t[j])==1)      /* read in the waveforms vd,vg,id, and the time */
{
 fprintf(f3,"%.4e\n",t[j]);
 fscanf(f1,"%lf",&vd[j]);
 fscanf(f1,"%lf",&vg[j]);
```

```
 fscanf(f1,"%lf",&id[j]);
 j++;
}
    for (j=0; j < numstep ; j++)        /* at each timestep caclulate m,n,H=f(vgd,eox) */
{
 vgd[j]=(vg[j]-vd[j]);
 eox[j]=(vgd[j]-vfb)/tox;
 n[j]=ngd2*pow(eox[j],2.0)+ngd1*(eox[j])+ngdo;
 m[j]=mgd2*pow(eox[j],2.0)+mgd1*(eox[j])+mgdo;
 h[j]=pow(10,hgd2*pow(eox[j],2.0)+hgd1*(eox[j])+hgdo);
 {
  fprintf(f4,"%.9e \n",n[j]);
  fprintf(f5,"%.9e \n",n[j]);
 }
}
    c=0;
    while(fscanf(f2,"%lf",&ib[c])==1)  /* read in the Isub waveform into the array ib[c] */
{
 c++;
}
    fclose(f3);
    cumage(width[i],m,h,id,ib,numstep,vgd,im,t);/* call cumage to calculate age at each timestep numerically*/
    cage=fopen("nvscumage.dat","w"); /* open file "nvscumage.dat" to write n and cumulative age */
    f6=fopen("agei.dat","r");
    age=(double *)malloc(numstep*sizeof(double));
    cumlage=(double *)malloc(numstep*sizeof(double));
    for(z=1;z<numstep+1;z++)      /* write n and cumulative age in file cage*/
{
 if (z==1)
  {
   fscanf(f6,"%lf",&age[z-1]);
   cumlage[z-1]=age[z-1];
   fprintf(cage,"%e  %e \n",cumlage[z-1],n[z-1]);
  }
 else
  {
   fscanf(f6,"%lf",&age[z-1]);
   cumlage[z-1]=age[z-1]+cumlage[z-2];
   fprintf(cage,"%e  %e \n",cumlage[z-1],n[z-1]);
  }
}
    fclose(cage);
    fclose(f6);
    fclose(f4);
    fclose(f5);
    fclose(f1);
    fclose(f2);
    ageT=totage(0,numstep,numstep,age);
    agetot=ageT*cyc;                /* agetot is the total age of each device and tnum is transistor name*/
    fprintf(NOUT,"%s        ",tnum); /* olddeg calculates degradation&lifetime using old method */
    olddeg(n,age,numstep,BOUT,OLD,p,cyc,ldef,tnum,agetot,year);
                        /* newdeg calculates degradation & lifetime using new method */
                        /* maxdeg corresponds to the maximum degradation */
                        /* the file asym contains the AGE and n of all the asymptotes */
    newdeg(tnum,cyc,p,numstep,n,age,&maxdeg,&maxage,&maxn,asym);
    fprintf(asym,"%s   Maxage   Maxn\n",tnum);
    fprintf(asym,"     %e       %e \n",maxage,maxn);
    /* lifetime calculates lifetime of each device using the new method */
    lifetime(n,age,ldef,numstep,&minage,&minn,&minlif);
```

```
    minlif=(minlif*p*1e-9)/(365*24*3600);
    fprintf(NOUT,"%e %e %e \n",minlif,maxdeg,agetot);
    minlif=1e99; /* reset the minlif to a big number and maxdeg to a small number for the next device */
    maxdeg=0.0;
   }
 fclose(NOUT);
 fclose(BOUT);
 fclose(asym);
}
int newdeg(char tnum[],double cyc,double p,int numstep,double nt[],double age[],double * maxdeg,double * max-
age,double * maxn,FILE * asym)
{
 int index[100],count1,count2;
 double * none,* ageone,* ntwo,* agetwo;
 double mean,deg0,deg1,deg2,n1,n2,age1,age2,agen1,agen2,yr,ageT,agetot;
 double year;
 int flag=1,i,j,ct,points;
 struct agedata result; /* result is a structure that holds data of the splitted asymptotes */
 /* ageT is the total age in a period */
 ageT=totage(0,numstep,numstep,age);
 /* mean is the n average weighted by age ; findarea finds the area under the n vs AGE curve */
 mean=findarea(0,numstep,numstep,age,nt)/ageT;
 /* result holds the data of the splitted asymptote; partition does the splitting of the asymptotes */
 result=partition(nt,age,mean,numstep);

 deg0=100*pow((ageT)*(cyc),mean); /* deg0 is the degradation due to the original asymptote */

 if (deg0 > *maxdeg) /* assign maxdeg to deg0 if deg0 > maxdeg */
  {          /* maxage and maxn correspond to the maxdeg.. ie. maxdeg=(maxage*cycle)^maxn */
   *maxdeg=deg0;
   *maxage=ageT;
   *maxn=mean;
  }

 ct=result.ct;   /* ct is a counter used as an index */
 /* index is an array which contains the intersection points in the n vs. AGE plot which results when the
   the curve is partitioned using n average */
 for (i=0;i <= ct;i++)
  {
   index[i]=result.ints[i];
  }
   age1=result.age1;
   age2=result.age2;       /* calculate degradation due to each of the "twins",ie deg1,deg2 */
   n1=result.n1;
   n2=result.n2;
   deg0=100*pow((age1+age2)*(cyc),mean);
   deg1=100*pow((age1*(cyc)),n1);
   deg2=100*pow((age2*(cyc)),n2);
   fprintf(asym,"%e   %e   %e\n",age1+age2,mean,deg0);
   fprintf(asym,"%e   %e   %e\n",age1,n1,deg1);
   fprintf(asym,"%e   %e   %e\n",age2,n2,deg2);
   if (age1 < TOL || age2 < TOL)  /* if one of the ages is too small then exit */
{
;
}
   else
{
 /* if deg1 or deg2 is the maximum degradation then
   reassign the maxdeg,maxage,maxn */
```

```
 if (deg1 > deg2 && deg1 > deg0)
   {
    if (deg1 > *maxdeg)
{
 *maxdeg=deg1;
 *maxage=age1;
 *maxn=n1;
}
   }
 if (deg2 > deg1 && deg2 > deg0)
   {
    if (deg2 > *maxdeg)
{
 *maxdeg=deg2;
 *maxage=age2;
 *maxn=n2;
}
   }
 count1=0;
 count2=0;      /* setup two arrays of age and n to do the splitting of each asymptote */
 none=(double *)malloc(numstep*sizeof(double));
 ntwo=(double *)malloc(numstep*sizeof(double));
 ageone=(double *) malloc(numstep*sizeof(double));
 agetwo=(double *) malloc(numstep*sizeof(double));
 for (i=0; i < ct; i++)
   {
    for (j=index[i];j<index[i+1];j++)
{
 if (result.f)
   {
    none[count1]=nt[j];
    ageone[count1]=age[j];
    count1++;
   }
 else
   {
    ntwo[count2]=nt[j];
    agetwo[count2]=age[j];
    count2++;
   }
}
    result.f=!(result.f);
   }
 /* split the "twins" by invoking newdeg twice ; one for each asymptote */
 newdeg(tnum,cyc,p,count1,none,ageone,maxdeg,maxage,maxn,asym);
 newdeg(tnum,cyc,p,count2,ntwo,agetwo,maxdeg,maxage,maxn,asym);

}

    return 1;
}


/* olddeg calculates the degradation using the old method ie. deg=aeff*t^neff */
void olddeg(double nt[],double age[],int numstep,FILE * BOUT,FILE * OLD,double p,double cyc,double ldef,char *
tnum,double agetot,double yr)
{
 int points,i;
 double * dg,* time,meanx,meany,slope,dcyc1,dcyc2,aeff,neff,ageeff,deg,berttime;
```

```
 double sumxy=0,sumy=0, sumxsq=0,sumx=0,b;


 points=2;  /* points defines how many points to use to fit the simulated data */
 dg=(double *)malloc(points*sizeof(double)); /* dg is an array that holds the simulated degradation data */
 time=(double *)malloc(points*sizeof(double));/* time is an array that holds time of the simulated data */
 for(i=0;i<points;i++)  /* findeg finds the degradation at the ith cycle using the iterative method */
  {
   dg[i]=finddeg(nt,age,numstep,i+1);
   dg[i]=log(dg[i]);
   time[i]=log(i+1);
  }
 /* the calculation below determines the fitting of the simulation points to determine aeff, neff for
   the old method */
 for(i=0;i<points;i++)
  {
   sumxy+=time[i]*dg[i];
   sumx+=time[i];
   sumy+=dg[i];
   sumxsq+=pow(time[i],2.0);
  }
 meanx=sumx/points;
 meany=sumy/points;
 slope=(points*sumxy-(sumx*sumy))/(points*sumxsq-pow(sumx,2.0));
 b=meany-slope*meanx;
 dcyc1=100*pow(2.718,(b+slope*log(1)));
 dcyc2=100*pow(2.718,(b+slope*log(2)));
 aeff=pow(2.718,b)/(pow(p*1e-9,slope));
 neff=slope;
 ageeff=pow(pow(2.718,b),1/slope);  /* calculating ageeff for the old method */
 deg=100*pow(2.718,(b+slope*log(cyc))); /* deg is the degradation at the specified point(cyc) */
 /* berttime is the lifetime from the old method */
 /* the result from the old method is stored in the file OLD  and BOUT */
 berttime=(pow(2.718,(log(ldef/100)-b)/slope))*p*1e-9/(365*24*3600);
 fprintf(BOUT,"%s        %e  %e  %e \n",tnum,berttime,deg,agetot);
 fprintf(OLD,"%s%s\n","Bert result  Transistor ",tnum);
 fprintf(OLD,"%s\n","Aeff      neff");
 fprintf(OLD,"%8e  ",aeff);
 fprintf(OLD,"%8e\n\n\n",neff);
 fprintf(OLD,"%s  %e \n\n","Effective age = ",ageeff);
 fprintf(OLD,"%s  %e \n","Degradation after 1 cycle = ",dcyc1);
 fprintf(OLD,"%s  %e \n","Degradation after 2 cycle = ",dcyc2);
 fprintf(OLD,"%s  %lf  %s  = %.2e \n","Degradation(%) at ",yr,"yrs.",deg);
 fprintf(OLD,"%s  %lf %s %.2e %s\n\n","Lifetime (",ldef,"%) =",berttime,"yrs. ");
}

/* function cumage calculates the age at each timestep and stores it in a file agei.dat and age.dat */
void cumage(double w,double m[],double h[],double id[],double ib[],int numstep,double vgd[],char im[],double t[])
{
 int c=0;
 double d,age0,age1,agei,cmage=0,dt;
 FILE * f1,*f2,*f3,*f4,*f5;
 int i,ct;
 f4=fopen("agei.dat","w");
 f5=fopen("age.dat","w");
 i=0;
 if (strcmp(im,"r")==0) /* im determines the integration method ; im="r" => rectangular im="t"=> triangular */
  {
   for(i=0;i<numstep-1;i++)
```

```
{
 dt = (t[i+1]-t[i]);
 if (fabs(ib[i])>0.0)
   {
    age0=2*pow(fabs(ib[i]/(id[i])),(m[i]))*(fabs(id[i])/(w*1e-6*h[i]));
    age1=0;
    agei=(age0+age1)*dt/2;
    fprintf(f4,"%e\n",agei);
    fprintf(f5,"%e\n",agei);
   }
 else
   {
    age0=0;
    age1=0;
    agei=(age0+age1)*dt/2;
    fprintf(f4,"%e\n",agei);
    fprintf(f5,"%e\n",agei);
   }
}
   }
 else /* triangular method for calculating age */
   {
    for (i=0;i<numstep-1;i++)
{
 dt = (t[i+1]-t[i]);
 if (fabs(ib[i])>0.0)
   {
    age0=pow(fabs(ib[i]/(id[i])),(m[i]))*(fabs(id[i])/(w*1e-6*h[i]));
    if (fabs(ib[i+1])>0.0)
{
 age1=pow(fabs(ib[i+1]/(id[i+1])),(m[i+1]))*(fabs(id[i+1])/(w*1e-6*h[i+1]));
 agei=(age0+age1)*dt/2;

 fprintf(f4,"%e\n",agei);
 fprintf(f5,"%e\n",agei);
}
    else
{
 age1=0;
 agei=(age0+age1)*dt/2;

 fprintf(f4,"%e\n",agei);
 fprintf(f5,"%e\n",agei);
}
   }
 else
   {
    age0=0;
    if (fabs(ib[i+1])>0.0)
{
 age1=pow(fabs(ib[i+1]/(id[i+1])),(m[i+1]))*(fabs(id[i+1])/(w*1e-6*h[i+1]));
}
    else
age1=0;
    agei=(age0+age1)*dt/2;

    fprintf(f4,"%e\n",agei);
    fprintf(f5,"%e\n",agei);
   }
```

```
 c++;
}
   }
 fprintf(f4,"%e\n",0.0);
 fprintf(f5,"%e\n",0.0);
 fclose(f4);
 fclose(f5);
}

/* partition is used to split the asymptote using the n average= mean  */
struct agedata partition(double n[],double age[],double mean,int size)
{
 FILE * file;
 int f;
 struct agedata result;
 double areaup=0,areab=0,ageup=0,nb=0,ageb=0,nup,ndown;
 int i,ct,count,flag=1,ind[100];
 ct=0;
 ind[ct]=0;
 ct++;
 /* partitioning routine ; ind is an array with the intersection points in the n vs. AGE plot which is
    partitioned with n=mean  */
 for(i=0;i<size;i++)
   {
    if (i==0 && n[i]<mean)
flag=0;
    if (i<(size-1))
{
 if (n[i]<=mean && n[i+1]>=mean)
   {
    if (n[i] != n[i+1])
{
 ind[ct]=i+1;
 ct++;
}
   }
 else
   if (n[i]>=mean && n[i+1]<=mean)
    {
if (n[i] != n[i+1])
 {
  ind[ct]=i+1;
  ct++;
 }
    }
}


   }
 f=flag;
 ind[ct]=size;/* the original asymptote is split into ageup and ageb and areaup, areab */
 /* areaup is the area in the n vs. AGE above the splitting line
    and areab is the area in the n vs. AGE below the splitting line ;
    ageup and ageb correspond to the ages of the "twins" */
 for (i=0;i<ct;i++)
   {
    if (flag)
{

 {
```

```
   areaup+=findarea(ind[i],ind[i+1],size,age,n);
   ageup+=totage(ind[i],ind[i+1],size,age);
  }
  flag=0;

}
    else
{
  {
   areab+=findarea(ind[i],ind[i+1],size,age,n);
   ageb+=totage(ind[i],ind[i+1],size,age);
  }
  flag=1;
}

   }
  /* naverage is the area under the n vs. AGE divided by the respective age  */
  /* result stores the data corresponding to the "twins " */
  nup=areaup/(ageup);
  ndown=areab/(ageb);
  result.n1=nup;
  result.age1=ageup;
  result.n2=ndown;
  result.age2=ageb;
  result.ct=ct;
  result.f=f;
  for (i=0;i<=ct;i++)
   {
    result.ints[i]=ind[i];
   }

  return result;
}
/* findarea finds the area under the n vs. AGE curve */
double findarea(int a,int b,int numstep,double age[],double n[])
{
  int i;
  return integral(a,b,n,age);
}

/* totage finds the total age between index a and index b */
double totage(int a,int b,int numstep,double age[])
{
  int i;
  double total=0;
  i=0;
  if (a==b)
   b++;
  for (i = a ; i<b; i++)
   {
    total+=age[i];
   }
  return total;
}
/* integrates the n vs. AGE curve numerically */
double integral(int a,int b,double n[],double age[])
{
  int i,j,count;
  double integrand=0;
```

```
 if (a==b)
   b++;
 for(i=a;i<b;i++)
   {
    integrand+=n[i]*age[i];
   }
 return integrand;

}

/* finddeg determines the degradation at cyc using the iterative method */
double finddeg(double n[],double age[], int count,int cyc)
{
 FILE * f;
 int i;
 double deg=0,aget=0;
 aget=0;
 for (i=0; i<(count*cyc); i++)
   {
    if (age[i%count] > TOL)
{
 aget=pow(deg,1/n[i%count]);
 aget+=age[i%count];
 deg=pow(aget,n[i%count]);
}

   }
 return deg;
}
/* filter reads the SPICE and BERT input and converts it so that it can readable by the
   program. sp is an array of files that stores the SPICE output(vg,vd,id) for each transistor
   and ibs is an array of files that store the BERT output(Isub)  */
int filter(int * num)
{
 FILE * f1,* f2,* spice[16],* isub[16],* spin,* wfile;
 char
sp[16][10]={"f1.dat","f2.dat","f3.dat","f4.dat","f5.dat","f6.dat","f7.dat","f8.dat","f9.dat","f10.dat","f11.dat","f12.dat"
,"f13.dat","f14.dat","f15.dat","f16.dat"};
 char
ibs[16][10]={"ib1.dat","ib2.dat","ib3.dat","ib4.dat","ib5.dat","ib6.dat","ib7.dat","ib8.dat","ib9.dat","ib10.dat","ib11.d
at","ib12.dat","ib13.dat","ib14.dat","ib15.dat","ib16.dat"};

 char
name2[16][20]={"ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(
","ISUB(","ISUB(","ISUB(","ISUB(","ISUB(","ISUB("};
 int position,numstep=0;
 double * w;
 char temp[50];
 char spiceout[30],spicein[30],isubfile[30];
 char index[20],time[20],ib[20];
 char line[]="--------------------------------------------------------------------------";
 int i,flag=0,count=0,end=1;
 char a[200],b[]="x",c[]="y";
 char tnum[10],t[10],vd[10],vg[10],id[10];
 char input[30],out[30],trname[16][10];
 printf("enter name of spice output file ");
 scanf("%s",spiceout);
 printf("enter name of substrate current file ");
 scanf("%s",isubfile);
```

```
 printf("enter the name of spice input file ");
 scanf("%s",spicein);
 f1=fopen(spiceout,"r");
 f2=fopen(isubfile,"r");
 spin=fopen(spicein,"r");
 while(fscanf(f1,"%s",&a)==1 && strcmp(a,b)!=0)
  {
  }
 for (i=0;i<MAXT;i++)
    {
spice[i]=fopen(sp[i],"w");
isub[i]=fopen(ibs[i],"w");
    }
 fscanf(f1,"%s%s%s%s",t,vd,vg,id);
 fscanf(f1,"%s%s%s%s",t,vd,vg,id);
 fprintf(spice[0],"%10s\n",id);
 strcpy(trname[0],id);
 count=0;
 strcpy(temp,"ISUB(");
 strcat(temp,id);
 strcat(temp,")");
 strcpy(name2[count],temp);
 flag=1;
 while(flag)
  {
   fscanf(f1,"%s%s%s%s",t,vd,vg,id);
   if(strcmp(t,c)==0)
{

 if (strcmp(vd,b)!=0)
  {
   count++;
   flag=0;
  }
 else
  {
   count++;
   strcpy(temp,"ISUB(");
   fscanf(f1,"%s%s%s%s%s%s",t,t,t,t,t,t);
   fprintf(spice[count],"%10s\n",t);
   strcpy(trname[count],t);

   strcat(temp,t);
   strcat(temp,")");
   strcpy(name2[count],temp);
   fscanf(f1,"%s%s%s%s",t,vd,vg,id);
  }
}
   if (flag)
fprintf(spice[count],"%20s%20s%20s%20s\n",t,vd,vg,id);
  }
 end=1;
 (* num)=count;

 count=0;
 flag=1;
 while(fscanf(f2,"%s",t)==1 && strcmp(t,"0")!=0)
  {
   for (i=0;i<(*num);i++)
```

```
{
 if (strcmp(t,name2[i])==0)
   position=i;
}
   }
 while(end)
   {
    fscanf(f2,"%s%s%s",time,ib,index);
    if (strcmp(index,line)==0)
{
 fprintf(isub[position],"%s\n",ib);
 numstep++;
 count++;

 if (count<(*num))
   {
    numstep=0;
    while(fscanf(f2,"%s",t)==1 && strcmp(t,"0")!=0)
{
 for (i=0;i<(*num);i++)
   {
    if (strcmp(t,name2[i])==0)
position=i;
   }
}
   }
 else
   end=0;
}
    else
{
 numstep++;
 fprintf(isub[position],"%s\n",ib);
}
   }
 fclose(f1);
 fclose(f2);
 count=0;
 wfile=fopen("width.dat","w");
 while(count < (*num))
   {
    fscanf(spin,"%s",t);
    if (compare(t,trname,(* num))==1)
{
 fscanf(spin,"%s %s %s %s %s %s",t,t,t,t,t,t);
 fwidth(t,wfile);
 count++;
}
   }

 for (count=0;count < (*num);count++)
   {
    fclose(spice[count]);
    fclose(isub[count]);
   }
 fclose(spin);
 fclose(wfile);
 return numstep;   /* filter function returns the number of timesteps in the analysis */
}
```

```
/* compare is used by filter for creating the files */
int compare(char name[],char trname[][10],int count)
{
 int i;

 for (i=0; i<count; i++)
  {

   if (strcmp(name,trname[i])==0)
return 1;
  }
 return 0;
}
/* fwidth returns the width of a device from the SPICE input netlist */
char * fwidth(char t[],FILE * wfile)
{
 int i,size;
 char * result;
 size=strlen(t)-3;
 result=(char *)malloc(size*sizeof(char));

 for(i=2;t[i] != 'u' && t[i] != 'U';i++)
  {
   result[i-2]=t[i];
  }
 result[strlen(t)-3]=t[strlen(t)];
 fprintf(wfile,"%s\n",result);
}
/* lifetime determines the lifetime of a device using the new method */
/* minlif is the initial guess to the lifetime; minage and minn correspond to the minlif */
double lifetime(double n[],double age[],double ldef,int numstep,double * minage,
double * minn,double * minlif)
{
 int index[100],count1,count2;
 double * none,* ageone,* ntwo,* agetwo;
 double mean,l0,l1,l2,n1,n2,age1,age2,ageT;
 int flag=1,i,j,ct;
 struct agedata result;
 ageT=totage(0,numstep,numstep,age); /* ageT is total age in a period */
 mean=findarea(0,numstep,numstep,age,n)/ageT;/* mean is the naverage weighted by age */
 result=partition(n,age,mean,numstep); /* partition initial asymptote */
 l0=pow(ldef/100,1/mean)/ageT; /* l0 is the lifetime predicted by the original asymptote */
 if (l0 < *minlif) /* reassign minlif if the l0 is smaller then the current lifetime(minlif) */
  {
   *minlif=l0;
   *minage=ageT;
   *minn=mean;
  }
 ct=result.ct;
 for (i=0;i <= ct;i++)/* index is the array with the intersection points of the n vs. AGE plot */
  {
   index[i]=result.ints[i];
  }
 age1=result.age1; /* age1,n1,age2,n2 are the result of splitting the original asymptote */
 age2=result.age2;
 n1=result.n1;
 n2=result.n2;
 l1=pow(ldef/100,1/n1)/age1; /* l1 and l2 are the lifetime using the two new asymptotes */
 l2=pow(ldef/100,1/n2)/age2;
```

```
 if (age1 < TOL || age2 < TOL) /* if any of the ages are small then exit */
   {
    ;
   }
 else
   {  /* if l1 or l2 predict a smaller lifetime then reassign the minlife,minage,minn */
    if (l1 < l2 && l1 < l0)
{
 if (l1 < *minlif)
   {
    *minlif=l1;
    *minage=age1;
    *minn=n1;
   }
}
    if (l2 < l1 && l2 < l0)
{
 if (l2 < *minlif)
   {
    *minlif=l2;
    *minage=age2;
    *minn=n2;
   }
}
    count1=0;
    count2=0;   /* create two sets of arrays of the "twins" for splitting */
    none=(double *)malloc(numstep*sizeof(double));
    ntwo=(double *)malloc(numstep*sizeof(double));
    ageone=(double *) malloc(numstep*sizeof(double));
    agetwo=(double *) malloc(numstep*sizeof(double));
    for (i=0; i < ct; i++)
   {
    for (j=index[i];j<index[i+1];j++)
{
 if (result.f)
   {
    none[count1]=n[j];
    ageone[count1]=age[j];
    count1++;
   }
 else
   {
    ntwo[count2]=n[j];
    agetwo[count2]=age[j];
    count2++;
   }
}
    result.f=!result.f;
   }
   /* split each of "twins" to determine the smallest lifetime */
 lifetime(none,ageone,ldef,count1,minage,minn,minlif);
 lifetime(ntwo,agetwo,ldef,count2,minage,minn,minlif);
}
    return 1;
}
```

# References

[1] Chenming Hu, Simon C. Tam, Fu-chieh Hsu, Ping-Keung Ko, Tung-Yi Chan, and Kyle W. Terrill, "Hot-electron-Induced MOSFET Degradation-Model, Monitor, and Improvement ," IEEE Transaction on Electron Devices, Vol. Ed-32, No.2, pp. 375-385, February 1985.

[2] Peter Maurice Lee, "Modeling and Simulation of Hot-carrier Effects in MOS Devices and Circuits", Ph.D. Thesis, 1990.

[3] Mary M. Kuo, Koichi Seki, Peter M. Lee, Jeong Yfol Choi, Ping K. Ko, and Chenming Hu, "Simulation of MOSFET Lifetime under AC Hot-electron Stress", IEEE Transaction on Electron Devices, Vol. 35, No. 7, pp. 1004-1005, July 1988.

[4] SeokWon A. Kim, Beniyam Menberu, Thomas E. Kopley, and James E. Chung, "Oxide-Field Dependence of the NMOS Hot-carrier Degradation Rate and its Impact on AC-Lifetime Prediction". In Proc. IEDM, December 1995, Washington D.C. .

[5] BTA BERT's User's manual, 1994

[6] SeokWon A. Kim, Beniyam Menberu, and James E. Chung, "A New Algorithm for NMOS AC Hot-Carrier Lifetime Prediction Based on the Dominant Degradation Asymptote", To appear IRPS April 1996.

[7] Lerman, Steven R. Problem Solving and Computation for Scientist and Engineers, 1993

[8] HSPICE User's Manual, Analysis and Methods, Meta-Software, Volume 3, 1992.

[9] Charles Sodini, and P.K. Ko, "The effect of high-fields on MOS device and circuit performance," IEEE Transaction on Electron Devices, Vol. ED-31, pp. 1386-1393, October 1984