Circuit-Level Reliability Simulator for Front-End-of-Line and Middle-of-Line Time-Dependent Dielectric Breakdown in FinFET Technology

Kexin Yang, Taizhi Liu, Rui Zhang, Linda Milor
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA USA
kyang70@gatech.edu

***Abstract*— This paper presents a lifetime simulator for both Front-End-of-Line (FEOL) time dependent dielectric breakdown (TDDB) and the newly emerging Middle-of-Line (MOL) time dependent dielectric breakdown for FinFET technology. A lifetime assessment flow for digital circuits and microprocessors is proposed for the target wearout mechanisms, and its associated vulnerable feature extraction algorithms are discussed in detail. Our simulator incorporates the detailed electrical stress, temperature, linewidth of each standard cell within the digital circuit and microprocessor. Also, FEOL TDDB and MOL TDDB lifetimes are combined in the calculation of TDDB lifetime. Circuit designers can use the resulting lifetime information to guide and improve their circuits to make them more robust and reliable way.**

Keywords—time-dependent dielectric breakdown; lifetime simulator; wearout; frontend-of-line dielectric breakdown; middle-of-line breakdown; digital circuit; microprocessor; reliability

I. INTRODUCTION

Traditional FEOL time-dependent dielectric breakdown (FEOL TDDB) is one of the main concerns for advanced CMOS technology. Accurate circuit lifetime assessment due to TDDB has become an significant part of the circuit design process. A new source of breakdown is Middle-of-Line (MOL) dielectric breakdown, which is breakdown between the polysilicon/high-k control gate (PC) and diffusion contacts (CA) [1]. MOL TDDB is a growing concern for semiconductor device reliability. It is necessary to investigate and perform detailed lifetime analysis of MOL TDDB in state-of-art FinFET technology.

This paper presents a simulator that can be used to assess logic circuit lifetime due to not only the traditional FEOL TDDB, but also the above mentioned MOL TDDB in FinFET technology. There are studies of MOL TDDB on dielectric materials [2], [3]. A budget-based MOL reliability management in FinFET technology is proposed in [4]; however, the authors declare that voltage does not have a strong impact on MOL TDDB, and assume a fixed voltage for the lifetime calculation. The assumption may be broken by compact standard cell layout and frequently switching activity, where the voltage difference between two segments plays an important role. In addition, no study has been conducted to investigate the vulnerable feature extraction algorithms in MOL TDDB in both digital circuits and microprocessors.

In previous FEOL system-level reliability studies, researchers have studied bias temperature instability (BTI) [5], [6], hot carrier injection (HCI) [7], [8], FEOL TDDB [9], [10]. None have considered MOL TDDB at the system-level which involves the extraction of vulnerable features of MOL TDDB in a circuit layout. In addition, most of the studies are conducted with traditional CMOS technology and fail to consider the state-of-art FinFET technology.

A methodology to link device level wearout models of MOL TDDB and FEOL TDDB to circuit lifetime is introduced in this work. The MOL TDDB vulnerable features in a FinFET are presented. The corresponding algorithms to extract such vulnerable features in a standard cell are discussed in detail. Our simulator runs in three steps. First, it characterizes the standard cell library corresponding to a given FinFET technology library (generating vulnerable features for each standard cell). After that, the simulator combines the vulnerable features with cell activity and the temperature profile to calculate TDDB lifetime of standard cells. The last step combines the lifetime of vulnerable features caused by both FEOL and MOL TDDB.

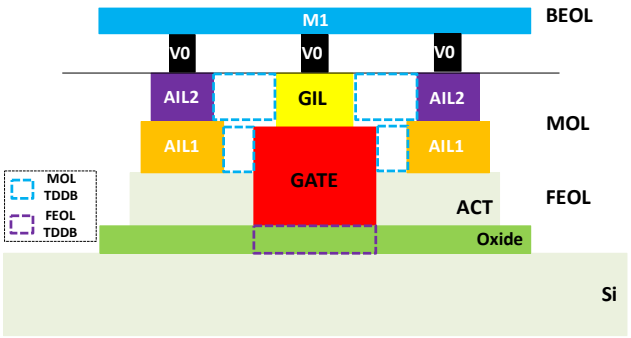


Fig. 1. FinFET cross-section.

To demonstrate our simulator's functionality, the lifetime simulator is used to study the lifetime distribution for an 8-bit FFT circuit and a Leon3 processor implemented with FinFET technology. For the Leon3, we also consider the impact of use scenarios on the lifetime of the microprocessor.

The rest of this paper is organized as follows. Section 2 presents the wearout models used for TDDB. Section 3 describes the extraction algorithm in detail. Section 4 presents the lifetime simulator using an FFT circuit and the Leon3 microprocessor as examples. The paper is concluded in Section 5.

II. DEVICE-LEVEL WEAROUT MODELS

Fig. 1 shows the breakdown paths of FEOL TDDB and MOL TDDB. FEOL TDDB is the breakdown between the gate and source or drain; whereas MOL TDDB is the breakdown between the gate and its adjacent contact or active interconnect layer.

A. FEOL TDDB Model

FEOL TDDB is described as the build-up of traps in the gate oxide as a function of time under voltage and thermal stress. We use the hard breakdown (HBD) model to characterize the transistor lifetime distribution. For ultra-thin (<5nm) gate dielectrics, the time-to-failure due to gate-oxide degradation can be derived by connecting the oxide degradation model to the Weibull failure distribution function [11] which is described by a shape parameter, β and a characteristic lifetime η, which is the time-to-failure at the 63% probability point, i.e.,

$$\eta = A_{ox}\left(\frac{1}{WL}\right)^{\frac{1}{\beta}} e^{-\frac{1}{\beta}} V^{a+bT} \exp\left(\frac{c}{T} + \frac{d}{T^2}\right) s^{-1} \quad (1)$$

where W and L are the device width and length, respectively, s is the probability of stress, T is temperature, V is gate voltage, and a, b, c, d, and A_{ox} are fitting parameters, which include the activation energy between 0.6 and 0.9 eV. The constants in (1) are determined using test structure data at high temperatures and voltages [9].

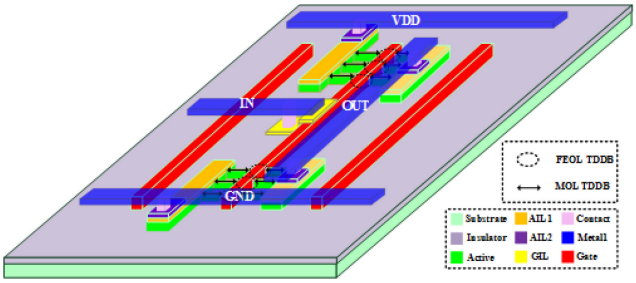


Fig. 2. 3D inverter view of MOL TDDB and FEOL TDDB.

B. MOL TDDB Model

Although Back-End-Line TDDB (BTDDB) is not discussed in this paper, the device-level lifetime model for MOL TDDB is similar to that of BTDDB [12] as follows:

$$\eta = A_{MTDDB} L_i^{-1/\beta_i} \exp(-\gamma E^m + E_a/kT) \quad (2)$$

where $A_{MOL\ TDDB}$ is a constant that depends on the material properties of the dielectric, γ is the field acceleration factor; electric filed is a function of voltage, V and the linespace, S_i, i.e., $E=V/S_i$, and m is 1 for the E model [13]. L_i is the vulnerable length and E_a is the activation energy (~0.5eV). The temperature dependence is modelled with the Arrhenius relationship [14], where k is the Boltzmann constant. The parameters are obtained from experimental data from [1], [14].

III. Vulnerable Feature Extraction

In this section, the algorithms that are used to extract vulnerable features in each standard cell for TDDB are introduced. As shown in Section 2, the device-level models for FEOL TDDB and MOL TDDB are different; thus, we need to develop a unique algorithm to extract vulnerable features due to each type of TDDB. In this study, FreePDK15 [15] was implemented and used as a case study for FinFET technology. In Fig. 1, a cross-section view of a FinFET transistor is presented. The blue dashed squares stand for locations for MOL TDDB, while FEOL TDDB is represented by the purple dashed squares. A detailed 3D illustration of TDDB in a layout in FinFET technology can be found in Fig. 2, which uses an inverter's layout as an example.

A. FEOL TDDB Vulnerable Feature Extraction

To characterize device's FEOL TDDB lifetime, we only need to obtain the transistor's width (W) and channel length (L). We can get the transistor's size information for each standard cell from the spice netlist. Notice instead of using the width (W) directly from the netlist, which represents the drawn width of the source and drain, we should calculate the effective width [16] as follows,

$$W_{eff} = T_{fin} + 2H_{fin} \quad (3)$$

where T_{fin} is the fin thickness and H_{fin} is the fin height.

To take the number of fins into account, we obtain the total effective width,

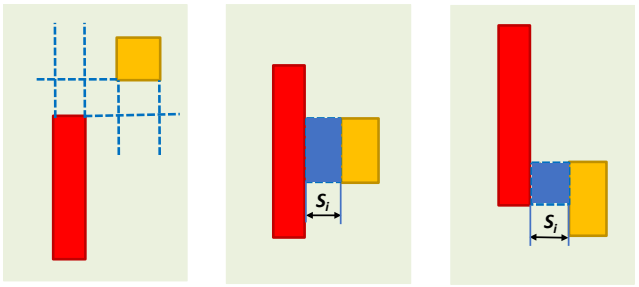
$$W_{eff,total} = n_{fin} \cdot W_{eff} \quad (4)$$

where n_{fin} is number of fins in the transistor.

B. MOL TDDB Vulnerable Feature Extraction

We need to analyze and extract MOL TDDB vulnerable features from each standard cell layout. From Fig 1, there are two types of MOL TDDB features; one is the GATE-AIL1 pair and the other is the GIL-AIL2 pair. Our simulator needs to find all the existing vulnerable features.

As shown in Fig 3, L_i and S_i are indicated. The yellow square represents the AIL1 layer; the red square stands for the gate. The blue dashed square is the vulnerable feature and our goal to extract these vulnerable features. In our simulator, only the nearest vulnerable features are extracted, since the further ones are separated by poly segments in the middle and the electric field between them will be shielded. The vulnerable features are



(a) (b) (c)

Fig. 3. Illustration of MOL TDDB vulnerable features.

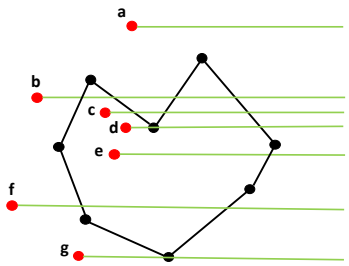


Fig. 4. Point inclusion problem.

divided into three categories. In Fig. 3(a), there is no overlap between a GATE-AIL1 pair. We call this the "nonoverlap" case (no vulnerable feature exists). In Fig. 3(b) and 3(c), when there are vulnerable features, we call this the "overlap" relationship. Fig 3(b) and 3(c) are the full "overlap" and partial "overlap" situations, respectively. A vulnerable feature only exists in the "overlap" situation, and thus, our algorithm will detect this situation.

For layout generation, we have used the NanGate 15nm Open Cell Library [17]. To identify the vulnerable features in a standard cell layout, we should find out the pin to which the corresponding segment (GATE or AIL1 layer) is connected. This is because the layout needs to be linked to the netlist and activity information for when the circuit is running benchmarks.

In the layout file, the pin connection is stored as a single point coordinate (x, y), while a polygon is stored as its vertices' coordinates: bottom left corner (*Left*, *Bottom*), and the upper right corner (*Right*, *Top*). Therefore, we need to start by finding the top layer to which the pin is directly connected and continue the process downward in the stack. In a standard cell, in most cases, the top layer will be a metal layer (M1 or M2) depending on the type of cell being analyzed.

Some layers are drawn as rectangle segments with four vertices, and the others are drawn as polygons which contain more than four vertices. The situation where there are irregular geometries adds complexity to our problem, and we introduce the point inclusion algorithm to determine the direct layer to which a pin is connected.

In our simulator, the ray-casting algorithm [18] has been implemented to find the connected layer. If the number of crossings is odd, the point is inside a polygon. Fig 4 gives a set of example points which we need to test. The Python implementation of the ray-casting algorithm to determine whether a point is inside a polygon is presented in Fig. 5.

After finding the pin's directly connected layer's segment, we start to process downward to find all the layers to which the pin is connected. There are two types of overlap we could find in a standard cell layout, which are illustrated in Fig 6. Fig 6(a) shows the overlap situation that only happens between GIL and GATE, while most of the overlap situations are presented in Fig 6(b).

The overlap between two rectangles is easy to implement and its corresponding algorithm is shown in Fig. 7. The proof of the "if statement" is by contradiction. Any one of the following four cases guarantee that no overlap exists between rectangles A and B:

Algorithm 1: point inclusion algorithm

Input: *polygon (poly), point (p)*
Output: whether the test point is inside the polygon
def PinPoly(*poly, p*):
  nvert = number of vertex in the polygon
  testx = p.x
  testy = p.y
  result = false
  i = 0
  j = nvert - 1
  while *(i < nvert):*
    if *(((poly.Vertex[i].y > testy) != (poly.Vertex[j].y > testy))*
    and *(testx < (poly.Vertex[j].x - poly.Vertex[i].x)*
    ** (testy-poly.Vertex[i].y) / (poly.Vertex[j].y-poly.Vertex[i].y)*
    + self.Vertex[i].x)):
      result = !result
    j = i
    i = i + 1
  return *result*

Fig. 5. Point inclusion algorithm.

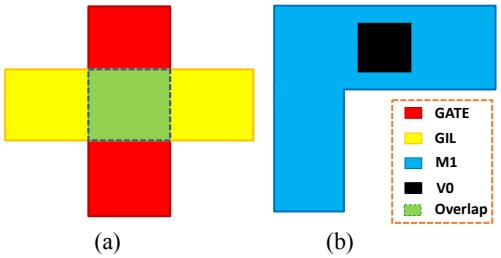


Fig. 6. Overlap of layer and via.

Algorithm 2: rectangle overlap determination

Input: rectangle_A (*RectA*), rectangle_B (*RectB*)
Output: whether the two rectangles overlap
if (*RectA.Left* < *RectB.Right* **and** *RectA.Right* > *RectB.Left* **and**
  RectA.Top > *RectB.Bottom* **and** *RectA.Bottom* < *RectB.Top*):
  return *true*
else:
  return *false*
end

Fig. 7. Overlap determination algorithm.

Case #1: If A's left edge is to the right of B's right edge (A is totally to the right of B).

Case #2: If A's right edge is to the left of B's left edge (A is totally to the left of B).

Case #3: If A's top edge is below B's bottom edge (A is totally below B).

Case #4: If A's bottom edge is above B's top edge (A is totally above B).

As for the second situation in Fig. 6(b), we can utilize the poly inclusion algorithm to do the job. That is, if one of the via's vertices is inside the polygon, then the two overlap. Once the layer connection determination is finished, we store the pin's connected layers for each standard cell in a Python dictionary named "*std_cell_info*".

As mentioned before, the cell layout is composed of multiple polygons with their corresponding vertices. To extract the vulnerable features in Fig. 8, we use the vulnerable feature extraction algorithm which is presented in Fig. 9.

We perform vulnerable feature extraction only if the "overlap" configuration of a GATE-AIL1 pair exists. If an GATE-AIL1 pair overlaps, the maximum y value of the bottom left corner of the GATE and AIL1, max(*G.Bottom*, *A.Bottom*)

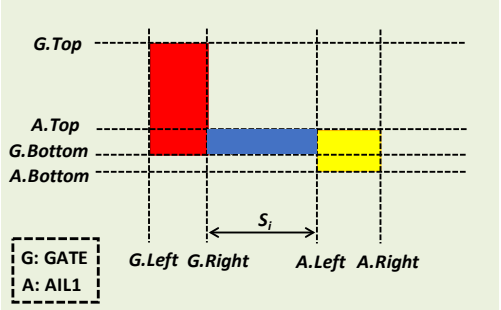


Fig. 8. Vulnerable feature extraction.

Algorithm 3: vulnerable feature extraction

Input: standard cell layout information (.txt file)
Output: vulnerable feature for each standard cell
for each *pin* in standard cell:
  for each *pin* in *std_cell_info*[*pin'*] (*pin* != *pin'*):
    if max(G.*Bottom*, *A.Bottom*) < min(G.*Top*, *A.Top*):
      Li = min(G.*Top*, *A.Top*) - max(G.*Bottom*, *A.Bottom*)
      X_coord = sort([G.*Left*, G.*Right*, *A.Left*, *A.Right*])
      Si = *X_coord*[2] - *X_coord*[1]
    end
  end
end
#*std_cell_info* is a dictionrary which stores each pin connected layers

Fig. 9. Vulnerable feature extraction algorithm.

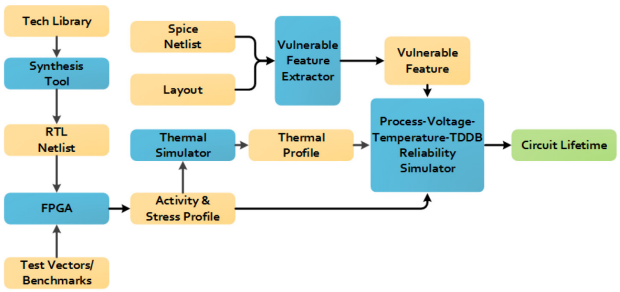


Fig. 10. Framework for the reliability simulator. Yellow boxes are data and blue boxes are tools.

must be less than the minimum y value of the upper right corner of GATE and AIL1, min(*G.Top*, *A.Top*).

The vulnerable length L_i is computed as follows,

$$L_i = \min(G.Top, A.Top) - \max(G.Bottom, A.Bottom) \quad (5)$$

To extract the linespace Si between the GATE-AIL1 pair, we sort the horizontal coordinates and put them into an array X_coord[] first. After sorting, the linespace can be easily computed by the subtraction of the middle two elements,

$$S_i = X\_coord[2] - X\_coord[1] \quad (6)$$

IV. Lifetime Simulator

The framework of our reliability simulator is presented in Fig. 10. This figure describes the tool flow needed to compute lifetime. The left most part of the figure includes the tools needed to determine operating profiles, such as activity, duty cycle, and temperature for each net, while the circuit is supplied with a set of random input vectors. The blocks on the right combine the operating profiles together to determine the lifetime. The lifetime is first computed for individual standard cells, and then these lifetimes are combined to find the lifetime of the whole circuit.

For FEOL TDDB and MOL TDDB, the significant factors are activity, voltage (VDD), temperature, and the vulnerable features. For activity tracking, the circuit netlist is loaded onto an FPGA for emulation [19]. The resulting state probabilities and toggle rates of the I/O ports are recorded. By using PrimeTime [20] activity propagation, we obtain the state probabilities and toggle rates for all the internal nets. The state probabilities are the key parameters to determine the lifetime of

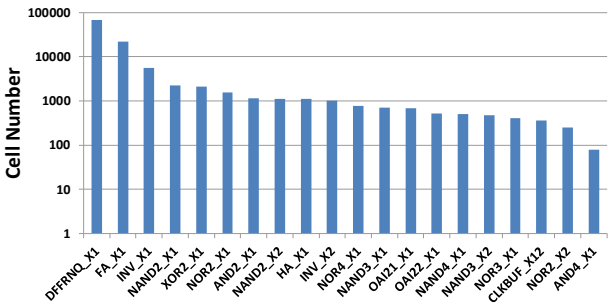


Fig. 11. FFT standard cell count.

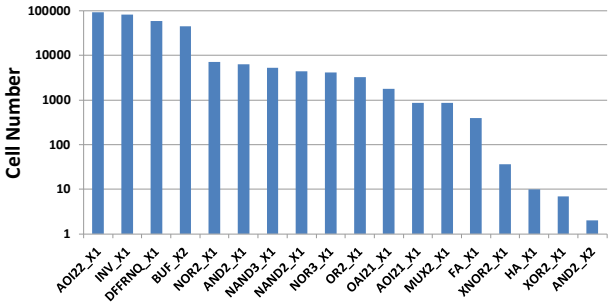


Fig. 12. Leon3 standard cell count.

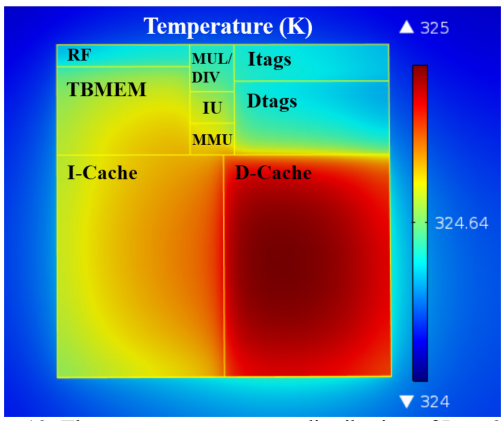


Fig. 13. The average temperature distribution of Leon3 while running a standard benchmark.

each layout feature, since signal states determine the time that each layout feature is under stress.

An 8-bit FFT circuit and a Leon3 microprocessor are implemented in FreePDK15 in this paper to demonstrate the functionality of our simulator. Synthesis is done with Synopsys Design Compiler [21], with the standard library with 69 different standard cells. After synthesis, the FFT circuit is composed of 38 types of standard cells and their corresponding top 20 cells in terms of their count are shown in Fig. 11. The Leon3 has 18 types of standard cells, and Fig. 12 shows the cell count.

Using the net activity and RC information from the layout, we can find the power consumed by each component of the Leon3 microprocessor. To determine the thermal distribution, we consider the self-heating effects of FinFETs [22] and supply the power consumption data to COMSOL. The temperature distribution when Leon3 is running a standard benchmark is shown in Fig. 13. We associate this temperature profile with every standard cell in the microprocessor to calculate lifetime.

The FFT circuit is supplied with randomly generated inputs and the circuit continuously performs the Fast Fourier Transformation on the data. For the Leon3, we consider the degradation under different use scenarios, as shown in Fig. 14. The use scenarios have different fractions of time when the system is in three modes: operation, standby, and off [23]. The activity profiles during the operation mode are determined by running benchmarks [24]. Our experimental results use a combination of standard benchmarks.

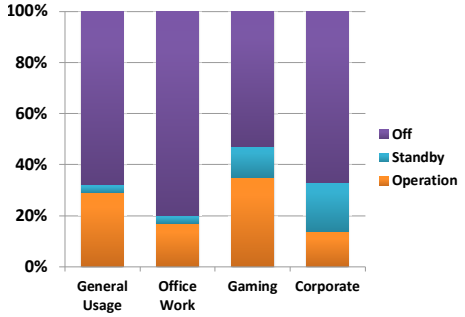


Fig. 14. The use scenarios provided by Intel [23].

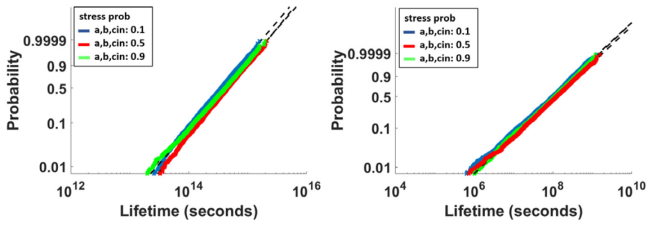


Fig. 15. Standard cell lifetime distribution for the FA (full adder) in the FFT circuit: (a) FEOL TDDB and (b) MOL TDDB.

First, we calculate the FEOL TDDB lifetime of a single device using (1) and the MOL TDDB lifetime of a single feature using (2). To combine different device lifetimes in a standard cell, we assume a standard cell is composed of n devices (n features for MOL TDDB), each modelled with a Weibull distribution, for each wearout mechanism. The characteristic lifetime of the cell, η_{cell}, is a combination of Weibull distributions and is the solution of [25]–[27]:

$$1 = \sum_{i=1}^{n} (\eta_{cell}/\eta_i)^{\beta_i} \quad (7)$$

where $\eta_i, i = 1, \ldots, n$ are the characteristic lifetimes of all of the devices; and $\beta_i, i = 1, \ldots, n$ are the corresponding shape parameters. Similarly [27]:

$$\beta_{cell} = \sum_{i=1}^{n} \beta_i (\eta_{cell}/\eta_i)^{\beta_i} \quad (8)$$

If the shape parameter is the same for each device (feature), which is typically assumed,

$$\eta_{cell} = \left(\sum_{i=1}^{n} \eta_i^{-\beta}\right)^{-1/\beta} \quad (9)$$

To calculate the FEOL TDDB lifetime of a standard cell, we need to obtain the gate-source voltage, V_{gs}, for each transistor and analyze each transistor's gate stress probability p. The lifetime of a transistor is a function of its stress probability, which in turn depends on the input pattern probabilities.

As for the MOL TDDB lifetime calculation, we should analyze the circuit's layout. For each adjacent GATE-AIL1 pair, we need extract the linespace S_i and vulnerable length L_i, as shown in Fig. 8. After that, for each GATE-AIL1 pair, the vulnerable feature pair (S_i, L_i) is associated with the poly-contact voltage difference V. The stress probability of a single dielectric segment feature is calculated as follows:

$$p_{total} = p_1 \cdot (1 - p_2) + p_2 \cdot (1 - p_1) \quad (10)$$

where p_1 and p_2 are the probabilities of the poly and contact being at logic "1", respectively.

By using (7) – (10), we get the characteristic lifetime of FEOL TDDB and MOL TDDB for every standard cell in the FFT circuit and the Leon3 microprocessor, which is shown in Figs. 15 and 16. The standard cell lifetime distributions are simply combinations of the transistor/layout feature lifetimes of all of the transistors and layout features in the standard cell. The input probabilities for each logic state for the standard cell propagate to internal nodes within the cell and determine the stress of transistors and layout features. As we can see in the figures, the lifetimes of the full adder cell and the OR gate are divided by the probability range for logic "1" at the inputs and plotted with a lognormal plot. We can partition the other

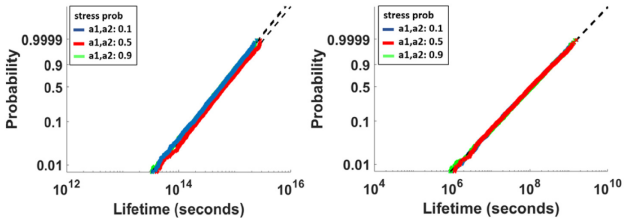


Fig. 16. Standard cell lifetime distribution for the OR2_X1 cell in Leon3: (a) FEOL TDDB and (b) MOL TDDB.

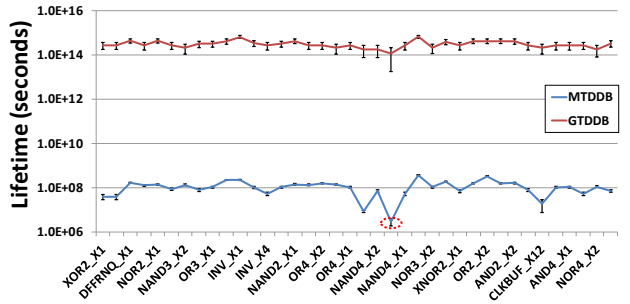


Fig. 17. FFT circuit FEOL TDDB and MOL TDDB characteristic lifetime for each type of standard cell and its lifetime limiting cell (shown in the red dashed circle). The confidence bound indicate variation in characteristic lifetime due to activity for each cell.

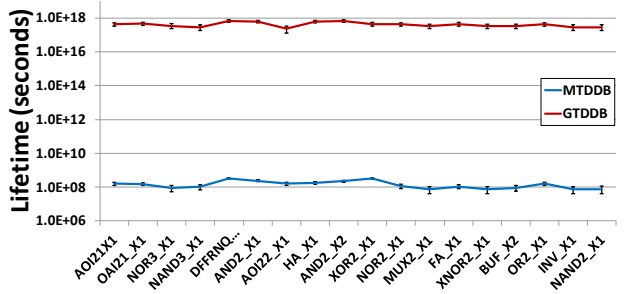


Fig. 18. Leon3 FEOL TDDB and MOL TDDB characteristic lifetime for each type of standard cell. The confidence bounds indicate variation in characteristic lifetime due to activity for each cell.

standard cells in the same way and fit their corresponding lifetimes with the lognormal distribution.

Regression is used to determine the parameters of the lognormal distribution. After regression, each standard cell is shown as a mean and standard deviation of the characteristic lifetime in Figs. 17 and 18.

Since the lifetime of each type of standard cell is calculated, we can proceed to calculate the circuit's failure probability at time t. We model each of the standard cell lifetimes with a lognormal distribution, for each wearout mechanism and its corresponding activity range. For n standard cells, there are n failure rates, F_i, $i = 1, \ldots, n$, where F_i is the cumulative probability of the lognormal distribution. These n failure rates contribute a reliability defect density d_i, which must be added together to find the failure probability of the whole circuit. The failure probability of the whole circuit F_{total} is a combination of lognormal distributions and is calculated as follow [28]:

$$d_i = -\ln(1 - F_i(t)) \quad (11)$$

$$F_{total} = 1 - exp^{-\sum d_i} \quad (12)$$

For each (μ_i, σ_i) pair, we can obtain a F_i by looking up the cumulative probability for a normal distribution. By using (11) and (12), we can combine standard cell lifetimes for FEOL TDDB and for MOL TDDB together, since they are independent mechanisms. The calculated failure probabilities are shown in Fig. 19.

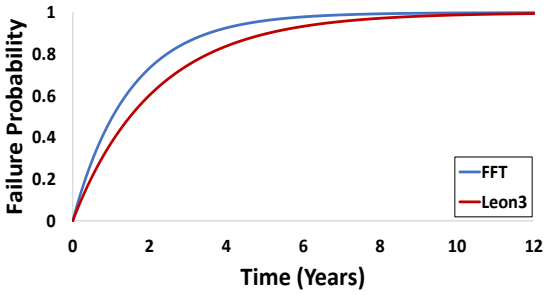


Fig. 19. FFT and Leon3 failure probability.

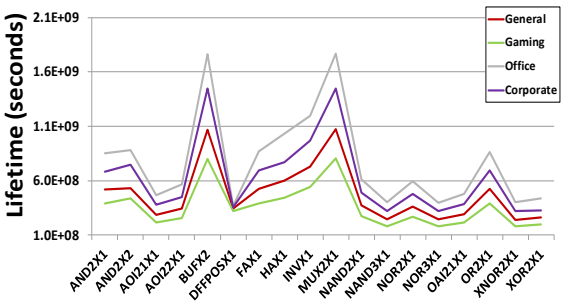


Fig. 20. Leon3 MOL TDDB lifetime for different use scenarios.

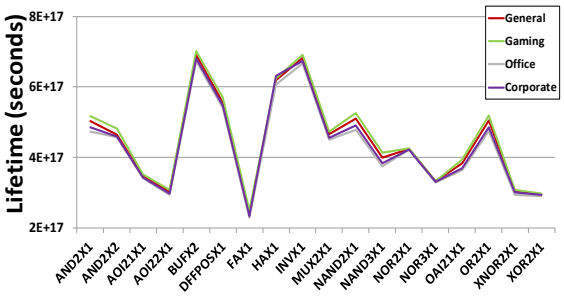


Fig. 21. Leon3 FEOL TDDB lifetime for different use scenarios.

From Figs. 17 and 18, we can see in the result that MOL TDDB is dominant. This result is technology dependent, since in FinFET technology, the layout is more compact, and thus the linespace and vulnerable length become smaller, which translates into a severe degradation due to MOL TDDB. On the other hand, FEOL TDDB is more sensitive to voltage, and the voltage scaling helps to alleviate the impact of FEOL TDDB.

To consider the use scenario impact on the Leon3, the stress during operation is computed based on activities when the Leon3 runs a standard benchmark. It idles with a random state in standby mode and powers down for the off mode. Shown in Fig. 20, we can conclude that MOL TDDB is more sensitive to use scenarios while, as observed in Fig. 21, FEOL TDDB is not sensitive. The vulnerable features in MOL TDDB are associated with two pins in a standard cell, while for FEOL TDDB, each device is only associated with its gate voltage; and thus, the disturbance of two pins will have a larger impact than just one.

In addition, one can find from Fig. 20 that not all standard cells have the same lifetime degradation under different scenarios; and thus, it is possible for a circuit designer to choose specific types of cell over others to ensure a longer lifetime in certain applications.

After assessing all the cell lifetimes in a circuit under different use scenarios; it is possible to replace the lifetime limiting standard cells with the ones with longer lifetime. As we can see in Fig. 17, for the FFT circuit, the "INV_X16" cell is the lifetime limiting cell in the FFT circuit, and thus, we could use 16 "INV_X1" cells to replace the "INV_X16" cell to achieve the same functionality with a higher lifetime. The layout of "INV_X16" is more compact and has more vulnerable features

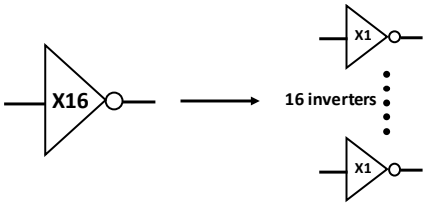


Fig. 22. Using cells with longer lifetime to replace a lifetime limiting cell to improve the lifetime of the circuit.

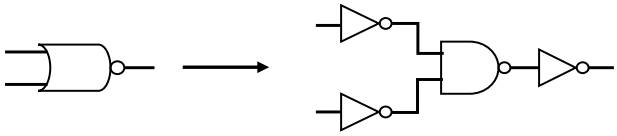


Fig. 23. Using a inverter and a NAND gate to replace a NOR.

in the layout, which causes a lower lifetime. An illustration is shown in Fig. 22; if the inverter is operated at 50% duty cycle, we can increase the lifetime by 6.75X while increasing the area by 2.67X. In addition, further optimization is possible if we characterize each standard cell in the technology library and find a combination of cells that has the same function with a longer lifetime, at the expense of possibly more area and power. Fig. 23 gives another example which uses a combination of cells to replace a NOR gate in the circuit.

V. Conclusion

This paper investigates not only the traditional reliability concern, FEOL TDDB, but also the newly emerging wearout mechanism, MOL TDDB. A novel lifetime simulator for FinFET technology is proposed for target wearout mechanisms. The shrinking feature size leads to severe degradation caused by MOL TDDB because of its sensitivity to alignment errors. On the other hand, the voltage scaling alleviates the impact of FEOL TDDB and MOL TDDB.

With reliability simulation, a circuit designer can use the information to redesign a circuit or redraw the layout in a more robust and reliable way; also, a circuit designer can use application specific information to choose certain cells that have longer lifetimes than others. It is also possible to use the lifetime information to add some constraints on circuits to ensure the circuit's performance over the product lifetime.

This work gives a framework to identify the lifetime limiting cell in a circuit; further optimization on trade-off between power, area and lifetime of circuit needs further investigation. Also, in this study only the FEOL TDDB and MOL TDDB are considered. Future work can add more wearout mechanisms, such as BTI and HCI in the frontend-of-line, together with the backend-of-line wearout mechanisms: backend TDDB, electromigration (EM) and stress induced voiding (SIV).

References

[1] F. Chen, Carole Graas, Michael Shinosky, Kai Zhao, Shreesh Narasimha, Xiao Hu Liu, Chunyan Tian, "Breakdown data generation and in-die deconvolution methodology to address BEOL and MOL dielectric breakdown challenges," Microelectronics Reliability, vol. 55, no. 12, pp. 2727–2747, 2015.

[2] F. Chen, Carole Graas, Michael Shinosky, Chuck Griffin, Roger Dufresne, Ronald Bolam, Cathryn Christiansen, Kai Zhao, Shreesh Narasimha, Chunyan Tian, Choon-Leong Lou, "New breakdown data generation and analytics methodology to address BEOL and MOL dielectric TDDB process development and technology qualification challenges," IEEE Int. Reliability Physics Symp., 2014, pp. 3A.1.1-3A.1.11.

[3] E. Wu, J. Stathis, B. Li, B. Linder, K. Zhao, and G. Bonilla, "A critical analysis of sampling-based reconstruction methodology for dielectric breakdown systems (BEOL/MOL/FEOL)," IEEE Int. Reliability Physics Symp., 2015, pp. 2A.2.1-2A.2.11, 2015.

[4] and J. C. Ahn, Jae-Gyung, Ming Feng Lu, Nitin Navale, Dawn Graves, Gamal Refai-Ahmed, Ping-Chin Yeh, "Product-level reliability estimator with budget-based reliability management in 16nm technology," IEEE Int. Reliability Physics Symp., 2017, pp. 3A–3.1–3A–3.6.

[5] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao, "The impact of NBTI effect on combinational circuit: modeling, simulation, and analysis," IEEE Trans. on Very Large Scale Integration Systems, vol. 18, no. 2, pp. 173–183, 2010.

[6] T. Liu, C.-C. Chen, and L. Milor, "Comprehensive Reliability-Aware Statistical Timing Analysis Using a Unified Gate-Delay Model for Microprocessors," IEEE Trans. on Emerging Topics in Computing, 2016.

[7] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," IEEE Trans. on Device and Materials Reliability, vol. 7, no. 4, pp. 509–517, 2007.

[8] Y. Wang, S. Cotofana, and L. Fang, "A unified aging model of NBTI and HCI degradation towards lifetime reliability management for nanoscale MOSFET circuits," IEEE/ACM Int. Symp. on Nanoscale Architectures, 2011, pp. 175–180.

[9] X. Li, J. Qin, and J. B. Bernstein, "Compact modeling of MOSFET wearout mechanisms for circuit-reliability simulation," IEEE Trans. on Device and Materials Reliability, vol. 8, no. 1, pp. 98–121, 2008.

[10] K. Yang and L. Milor, "Impact of stress acceleration on mixed-signal gate oxide lifetime," IEEE Int. Mixed-Signals Testing Workshop, 2015.

[11] C.-C. Chen and L. Milor, "System-level modeling and microprocessor reliability analysis for backend wearout mechanisms," Design, Automation & Test in Europe Conf. & Exhibition, 2013, pp. 1615–1620.

[12] G. S. Haase and J. W. McPherson, "Modeling of interconnect dielectric lifetime under stress conditions and new extrapolation methodologies for time-dependent dielectric breakdown," IEEE Int. Reliability Physics Symp., 2007, pp. 390–398.

[13] K.-Y. Yiang, H. W. Yao, and A. Marathe, "TDDB Kinetics and their Relationship with the E-and√ E-models," Int. Interconnect Technology Conf., 2008, pp. 168–170.

[14] T. Kauerauf, A. Branka, G. Sorrentino, P. Roussel, S. Demuynck, K. Croes, K. Mercha, J. Bommels, Z. Tokei, and G. Groeseneken, "Reliability of MOL local interconnects," IEEE Int. Reliability Physics Symp., 2013, pp. 2F.5.1-2F.5.5.

[15] NCSU, "FreePDK15." [Online]. Available: https://www.eda.ncsu.edu/wiki/FreePDK15:Contents. [Accessed: 01-Jan-2017].

[16] J.-W. Yang and J. G. Fossum, "On the feasibility of nanoscale triple-gate CMOS transistors," IEEE Trans. on Electron Devices, vol. 52, no. 6, pp. 1159–1164, 2005.

[17] NanGate, "NanGate FreePDK15 Open Cell Library." [Online]. Available: http://www.nangate.com/?page_id=2328. [Accessed: 01-Jan-2017].

[18] R. J. Segura and F. R. Feito, "An algorithm for determining intersection segment-polygon in 3D," Computers & Graphics, vol. 22, no. 5, pp. 587–592, 1998.

[19] C.-C. Chen, S. Cha, T. Liu, and L. Milor, "System-level modeling of microprocessor reliability degradation due to BTI and HCI," IEEE Int. Reliability Physics Symp., 2014, pp. CA.8.1-CA.8.9.

[20] "PrimeTime." [Online]. Available: https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html. [Accessed: 01-May-2017].

[21] "Design Compiler." [Online]. Available: https://www.synopsys.com/support/training/rtl-synthesis/design-compiler.html. [Accessed: 01-May-2017].

[22] "C.O.M.S.O.L. Multiphysics,Heat Transfer Module User's Guide Version 5.2, COMSOL," 2015.

{23] R. Kwasnick, A. E. Papathanasiou, M. Reilly, A. Rashid, B. Zaknoon, and J. Falk, "Determination of CPU use conditions," Int. Reliability Physics Symp., 2011, pp. 2C.3.1-2C.3.6.

[24] "Mibench benchmark." [Online]. Available: http://vhosts.eecs.umich.edu/mibench//. [Accessed: 01-May-2017].

[25] M. Bashir and L. Milor, "Towards a chip level reliability simulator for copper/low-k backend processes," Design, Automation and Test in Europe, 2010, pp. 279–282.

[26] M. Bashir, D. H. Kim, K. Athikulwongse, S. K. Lim, and L. Milor, "Backend low-k TDDB chip reliability simulator," Int. Reliability Physics Symp., 2011, pp. 2C.2.1-2C.2.10.

[27] M. Bashir, L. Milor, D. H. Kim, and S. K. Lim, "Methodology to determine the impact of linewidth variation on chip scale copper/low-k backend dielectric breakdown," Microelectronics Reliability, vol. 50, no. 9, pp. 1341–1346, 2010.

[28] L. Milor and C. Hong, "Area scaling for backend dielectric breakdown," IEEE Trans. on Semiconductor Manufacturing, vol. 23, no. 3, pp. 429–441, 2010.