Hardware Aging-Based Software Metering

Foad Dabiri and Miodrag Potkonjak
Computer Science Department
University of California Los Angeles
email: {dabiri, miodrag} @ cs.ucla.edu

***Abstract*—Reliable and verifiable hardware, software and content usage metering (HSCM) are of primary importance for wide segments of e-commerce including intellectual property and digital rights management. We have developed the first HSCM technique that employs intrinsic aging properties of components in modern and pending integrated circuits (ICs) to create the first self-enforceable HSCM approach. There are variety of hardware aging techniques that range from electro-migration in wires to slow-down of crystal-based clocks. We focus on transistor aging due to negative bias temperature instability (NBTI) effects where the delay of gates increases proportionally to usage times.**

We address the problem of how we can measure the amount of time a particular licensed software (LS) is used by designing an aging circuitry and exposing it to the unique inputs associated with each LS. If a particular LS is used longer than specified, it automatically disables itself. Our novel HSCM technique uses a multi-stage optimization problem of computing the delays of gates, their aging degradation factors, and finally LS usage using convex programming. The experimental results show not just viability of the technique but also surprisingly high accuracy in the presence of measurement noise and imperfect aging models. HSCM can be used for many other business and engineering applications such as power minimization, software evaluation, and processor design.

I. Introduction and Related Work

Annual economic losses due to software, semiconductor integrated circuits, and content piracy have surpassed the $100 billion level. Annual BSA and IDC Global Software Piracy Studies estimate that more than 1/3 of software installed in 2006 on personal computers worldwide was illegal resulting into almost $40 billion in revenue and similar profit losses due to software piracy [1]. Table 1 shows the annual losses due to software piracy for the last four years. KPMG and the Alliance for Gray Market and Counterfeit Abatement claim that gray market sales for integrated circuits (ICs) also account for $40 billion in lost revenue each year and reduce profits of IC manufacturers for almost $5 billion annually [2].

Our primary goal is not just to address software piracy problems using novel device aging-based security mechanisms, but to enable new pricing models where the software fees are functions of software use. The new techniques are generic in the sense that they are directly applicable to content metering. Hardware usage metering is a special case where all executed functionality is considered as a single software package. The economic and engineering ramifications of the new generation of security protocols are significantly far reaching since it is possible to design protocols for rapid zero knowledge authentication and secret key exchange.

TABLE I
Table 1: Losses due to worldwide software piracy trends (in billion dollars

Year	2003	2004	2005	2006
Losses ($B)	28.80	32.78	34.48	39.58

Computational security has been the traditional field of study for intellectual property management. IP protection such as software and hardware usage metering are among the problems studied in this field. Web page access metering has been addressed by a number of researchers and companies. For example, Pitkow proposed techniques to uniquely identify users and to compensate for the usage of proxies and cashes [9]. A new mechanism for metering the popularity of web-sites was proposed by Franklin and Malkhi [5]. Naor and Pinka's schemes measure the amount of service requested from servers by clients [8]. Licensing has been the most popular methods used for software protection among vendors.

Over $40 billion of installed third party software uses GLOBEtrotter's FLEX1m electronic commerce for software technology. Today's dominating software licensing mechanism is based on the license key concept. A key is encrypted by using a string of data that contains software package ID and its usage constraints (e.g. expiration date) and the serial number of the computer where the key is installed. The invocation of the software package is done automatically when software is invoked by using one of the password schemes [10].

Device aging is an irreversibly inherent process in essentially all integrated circuits (ICs) and system technologies. Electromigration impacts all tungsten contacts between transistors and wires and wires themselves. Transistor delay and power characteristics deteriorate as a consequence of hot-carrier-Induced (HCI) and Negative Bias Temperature Instability (NBTI) effects. As a consequence of transistor activity, its structure deteriorates following power laws. From the circuit designers' perspective, the NBTI degradation process manifests as an increase of device threshold voltage (V_{th}), which in turn results in the slowdown of transistor switching speed. Similar degradation has also been observed in n-MOSFET transistors but its effect is far less critical than NBTI on p-MOSFET and hence it is negligible [4][6] [7].

Our approach is built on creating and leveraging key connections: (i) the correlation between the switching activity stress on each gate and its delay increase; (ii) the correlation between the inputs to the IC and the stress on each gate; and

978-3-9810801-5-5/DATE09 © 2009 EDAA

(iii) integration of a finite-state-machine (FSM) and circuitry.

The key technical challenges of the new e-commerce security approach are: (i) creation of a circuitry that has the property that from the aging of its gates one can reconstruct how often each of a number of input vectors is applied; (ii) creation of the input to the circuitry for each software package (or dataset) that facilitate the reconstruction; (iii) extraction of the increase of the delay of each gate and calculation of corresponding time of usage for each gate in the presence of measurement and aging model errors; and (iv) development of hardware mechanisms that are resilient to physical and other security attacks for enforcing software digital right management.

The last task is already solved in recent papers [11][3] by connecting the circuitry to existing of new finite state machine (FSM). FSM allows the overall integrated circuit to operate only if it receives the anticipated output response from the circuitry. The third task is solved optimally and is described in this paper. The reminder of the paper is organized in the following way: Section II introduces the preliminaries for our method and in Section III, we demonstrate the high level approach in our technique followed by an illustrative example. Section IV covers the details of our novel software metering technique and we show its effectiveness through simulation results in Section V. Conclusions and future directions in hardware aging based software metering are summarized in Section VI.

II. Preliminaries

Negative bias temperature instability (NBTI) has become one of the major causes for performance degradation of nanoscale circuits. We use this intrinsic property to characterize degradation of digital circuits and utilize it for intellectual property management.

Modern digital circuits are composed of CMOS gates. In CMOS devices, the NBTI-induced threshold voltage shift will occur over time, depending on the operating conditions of the device. The interaction of inversion layer holes with hydrogen-passivated Si atoms can break the SiH bonds, creating an interface trap and one H atom that can diffuse away from the interface (through the oxide) or can anneal an existing trap. The interface trap generation is modeled successfully in the ReactionDiffusion framework [12].

Bias temperature stress under constant voltage (DC) causes the generation of interface traps (N_{IT}) between the gate oxide and silicon substrate, which translate to device threshold voltage (V_t) shift and loss of drive current (I_{on}). The NBTI effect is more severe for PMOS FETs than NMOS FETs due to the presence of holes in the PMOS inversion layer that are known to interact with the oxide states [6].

Before we introduce models used for NBTI effects on digital circuits, let's first review how gate delay is dependent on different parameters. The propagation delay of a CMOS based digital gate can be expressed as:

$$d = \frac{C_L V_{dd}}{I_d} = \frac{C_L V_{dd} L_{eff}}{\mu C_{ox} W_{eff} (V_{gs} - V_{th})^{\alpha}} \quad (1)$$

where α is the velocity saturation index, V_{dd} is the supply voltage, C_L contains the parasitic capacitance and other parameters are technology dependant constants.Using equation 1 the delay degradation, Δd, for a given gate can be derived as:

$$\frac{\Delta d}{d_0} = \frac{\alpha \Delta V_{th}}{V_{gs} - V_{th}} \quad (2)$$

where d_0 is the original delay of the gate without any V_{th} degradation, and can be extracted from third-party time analysis tools.

NBTI causes circuit aging which will introduce a shift in V_{th} over time. The shift in the transistor threshold voltage, ΔV_{th}, can be derived using analytical models at [6][4]. Now, the question is how does V_{th} degrade as a device is being used. There are several studies which cover this issue thoroughly and model the aging of digital circuits. In [6], an analytical model of NBTI degradation has been introduced which relates V_{th} degradation to usage time as follows:

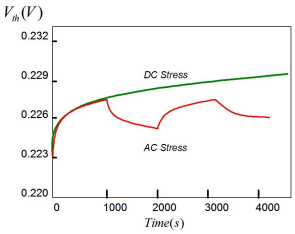

Fig. 1. V_{th} dependency on DC and AC stress as a function of stress time. This graph is taken from [11]

$$\Delta V_{th} = K_C \times \alpha_S S_i^{\frac{2}{3}} \times t^{\frac{1}{6}} \quad (3)$$

which illustrates the power dependency of V_{th} degradation with a fixed time exponent of $1/6$. Equation 3 will be the basis of our hardware-aging based software metering since it relates gate usage time (stress) to V_{th} shift. When a gate is being used it means that it is under either DC or AC stress. We use 'gate usage time' and 'stress time' interchangeably in this paper.

A. Problem Definition

The goal of IP protection under study in this paper is to perform hardware, software and component metering (HSCM) and particularly to determine how much a piece of software is used on a particular device. Assume we have a set of k software (applications, components...), $\Sigma = \{S_1, ..., S_k\}$, where each software S_i is run multiple times for an unknown arbitrary time t_i. The objective is to find the t_is efficiently with highest accuracy in the presence of measurement errors and imperfect degradation models.

III. Conceptual Basis

In this section, we describe the high level approach to software metering technique that we use for IP protection. The first step of the proposed method is to use a predesigned

circuitry which is embedded into a device and/or chip and is used as the aging circuitry for device characterization. This circuitry is called the *aging circuit* and is required to possess the following properties; the aging circuit is a especially structured circuit composed of logic gates which under NBTI age in a way that gate degradation can be measured effectively. Every software S_i is associated with a unique input vector ρ_i. Whenever that software is used, the corresponding input vector will be fed to the aging circuit and causes DC stress on a unique subset of the gates in the aging circuitry.

Figure 2 illustrates the NBTI-based aging effect on circuit characteristics and how we can use extract and use that information for software metering. The diagram on the left represents the high-level physical process of aging whereas the right diagram is the steps of utilizing the aging process for software metering.

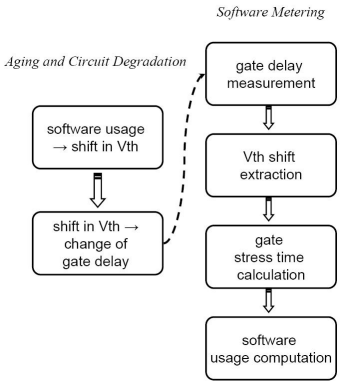

Fig. 2. The diagram on the left represents the high-level physical process of aging whereas the right diagram is the process of utilizing the aging process for software metering

At any given point of time, in order to measure software usage, the following steps are followed:

- Gate Delay Calculation: We measure the delay of several paths from the inputs of the aging circuit to its outputs. Path delays lead us to the calculation of individual gate delays using optimization techniques which will be covered in Section IV.
- Aging Factor Extraction: Once the delay of each gate is found, using the models in Section II we extract the degree to which each gate has been degraded, and therefore extract how long each individual gate has been under stress (usage).
- Software Metering: Each gate goes under stress for some set of software. Since software S_i has a unique signature vector ρ_i, it contributes to the aging of a subset of gates in the aging circuitry. Once the total usage of each gate is known, through another stage of optimization, individual execution (running) time of software is calculated.

To give more insight on our proposed software metering, we will illustrate the above procedure through a simple small example.

IV. Hardware, Software and Component Metering

Hardware Software and Component Metering (HSCM) is a multi-step process which involves aging circuit design/selection, signature vector generation, gate delay measurement, aging factor extraction and finally software metering. In this section, we will cover all these steps in detail to make HSCM an applicable process.

A. Aging Circuitry

In order to make the process of HSCM feasible, we use a predesigned circuit which enables accurate measurement of degradation and software/hardware usage. The aging circuitry is represented by a tuple $AG = (G, p, q)$ where $G = (V, E)$ is the directed graph representing the topology of the network and V and E are the sets representing the gates and connections (edges) in the circuit. Furthermore, p and q are input and output bits of the circuit respectively. For each gate $v_i \in V$ in the aging circuitry, there is a delay d_i associated with it. Depending on what inputs are fed into the aging circuitry, some of the gates will be under stress and experience aging and degradation caused by NBTI. As we discussed in Section II, this aging causes a shift in threshold voltage and eventually an increase in gate delay. The whole idea of HSCM is based on processing the changes in gate delays and extract software/hardware usage.

The selection of the aging circuitry is very important and can affect the HSCM process significantly. A 'good' aging circuit would be the one that can produce information usable for accurate software metering. Since all the information a circuit can give us is embedded inside gate characteristics, specially delay, we need to utilize a circuit which through standard methods of path delay measurements, individual gate delays can be calculated with highest degrees of accuracy even in the presence of measurement noise. Through the usage of path delays, individual gate delays can be extracted under the condition that there exist paths that are less-correlated and therefore inherit more entropy.

B. Input Vectors

An exact method to extract gate delays may require solving a linear system of equations of size $O(2^N)$, where N is the number of primary inputs. For large circuits with large numbers of primary inputs, the exact method is not computationally feasible. Therefore, we only use $|S|$ number of input configurations where $|S|$ is the number of software applications/components used on the device. Ideally, we would like to select input vectors such that the subset of gates under stress corresponding to each vector, enable us to pick as many paths as possible that are less-correlated and can be used to extract gate delays.

As we saw in Section III, for each software S_i, we assign a unique input vector ρ_i, called 'signature vector'. While a software S_i is being run on the system ρ_i will be fed constantly to the aging circuitry. This feeding causes DC stress to a subset of gates in the circuit and cause degradation and aging of the corresponding gates. For brevity, we don't cover the signature

vector creation process but there several techniques that fulfill the purpose such as m-of-n codes as a class of binary codes.

C. Gate Delay Characterization

The Digital Oscillation test is a known method for measuring path delays in digital circuits. In order to measure the propagation delay of a path, one input to a circuit is flipped and a change in the output is observed. This delay can be measured, and, with backtracking from the output to the input, the path is detected.

In this phase of our method, we measure and compute the delay of each gate in the aging circuitry using a set of path-delay measurements in the circuit. The delay of each path p_i from one primary input to one primary output is:

$$d_{p_i} = \sum_{\forall v_i \in p_i} d_{v_i} \tag{4}$$

where the sum is taken over all the gates in the path p_i. Let's assume that we generate m distinct path delays similar to equation 4. The set of m measurements can be presented as:

$$d_{p_i} = a_i^T x + v_i, \forall i; 1 \leq i \leq m \tag{5}$$

where $x \in \Re^n$is a vector of gate delays which is to be estimated, $d_{p_i} \in \Re$ is the measured path delay value and v_is are the measurement errors. We assume that the v_is are independent, identically distributed (IDD) with a normal distribution. In statistical estimation, a widely used method, called maximum likelihood (ML) estimation, is to estimate x as:

$$\hat{x}_{ml} = argmax_x p_x(d_p) = argmax_x l(x) \tag{6}$$

where $p_x(d_p)$ is the likelihood function of x and $l(x) = logp_x(d_p)$ is the log of the likelihood function which makes it easier to work with equation 6. Maximum likelihood estimation (MLE) is a popular statistical method used to calculate the best way of fitting a mathematical model to some data. Modeling real world data by estimating maximum likelihood offers a way of tuning the free parameters of the model to provide an optimum fit.

The likelihood function in this case is:

$$p_x(d_p) = \prod_{i=1}^{m} p(d_{p_i} - a_i^T x) \tag{7}$$

so the log-likelihood function would be:

$$l(x) = logp_x(d_p) = \sum_{i=1}^{m} log(p(d_{p_i} - a_i^T x)) \tag{8}$$

The ML estimate is any optimal point for the problem:

$$maximize : \sum_{i=1}^{m} log(p(d_{p_i} - a_i^T x)) \tag{9}$$

When v_i's are Gaussian with zero mean and variance σ^2 and density function calculate $p(z) = (2\pi\sigma^2)^{-\frac{1}{2}} e^{\frac{-v^2}{2\sigma^2}}$, the log-likelihood function would be:

$$l(x) = -(m/2)log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\|Ax - d_p\|_2^2 \tag{10}$$

where A is the matrix with rows $a_1^T, a_2^T, ..., a_m^T$. Therefore the maximum likelihood problem becomes equivalent to the solution of a least-square approximation which can be solved efficiently using convex programming.

D. Aging Factor Extraction and Software Usage Metering

The next step is to extract degradation factors and software usage. We divide this section into two: If the aging models in Section II are perfect or if the aging and degradation models include errors themselves.

1) Perfect Aging Models: The next step is to extract degradation factors and software usage. We divide this section into two parts: (i) If the aging models in Section II are perfect and (ii) if the aging and degradation models include errors themselves.

2) Perfect Aging Models: The gate delays measured in Section IV-C are used directly to determine individual gate degradation factor; for each gate v_i in the aging circuitry we have:

$$\Delta V_{th_i} = \frac{d_i - d_0}{d_0} \frac{V_{gs} - V_{th}}{\alpha} \tag{11}$$

$$t_i = \Theta(\Delta V_{th_i}) = (\frac{\Delta V_{th_i}}{K_C \times \alpha_S S_i^{\frac{2}{3}}})^6$$

$\Theta(\Delta V_{th_i})$ is the function wich maps threshold voltage change to usage time (stress time) of each gate.

The last phase of our HSCM techniques is the calculation of software usage. Let's assume that each software S_i is run for the total amount of τ_i. First of all, the total amount of each software's execution is important and not how that time has been broken over time. In other words, if a software is run for a period of time T_0, its effect on aging is equivalent to multiple runs which add up to T_0. Furthermore, the ordering of execution among other software is also irrelevant. In order to see why these clams are in fact true, remember that NBTI is a time dependant aging effect on digital circuits, and the total time that a gate is under stress is what causes the degradation.

Each and every software has caused stress on a specific subset of gates in the aging circuit through it's unique input vector. Let assume for each software S_i, the set of gates $\phi_i = \{v_{i_1}, .., v_{ik_i}\}$ are the gates which are under stress when software S_i is being run and k_i is the total number of gates which software S_i puts stress on. Therefore, for each arbitrary gate v_i, there is a set of software which causes stress on it, called $\varphi_i = \{S_{i_1}, .., S_{i_{r_i}}\}$, where r_i is the total number of software which cause stress on gate v_i.

In Section IV-D1, we measured the total stress time on each and every individual gate in the aging circuitry. In this phase of the HSCM, we extract the individual times that each software has been used. We form a linear programming formulation as follows: for each gate v_i, the total stress time t_i is equal to the total execution time of software which cause stress on v_i. In other words:

$$\sum_{j=0}^{r_i} \tau_j = t_i \tag{12}$$

where the sum is taken over all the execution times (τ_j) of software which cause stress on gate v_i. We construct the following LP formulation:

$$B\bar{\tau} = \bar{t} \tag{13}$$

where B is the coefficient matrix in which each row represent the coefficients in equation 12 and $\bar{\tau}$ and $\bar{t}$ are software usage times and gate stress times respectively.

The structure of the aging circuit and the fact that $|S| < |V|$ enables solving the above LP problem efficiently using classic LP solvers. The solution to equation 13 results in individual software usage times and finishes our HSCM method. As one can observe, many other variations can easily be adapted to our techniques. For instance, one other commonly important metric for IP protection and rights management is the number of times a particular software/hardware is used as opposed to the total usage times. In this scenario, the only modification is to feed the signature vector for constant duration of time, say t_c. Then, we follow the same procedure and at the end, by dividing τ_is by t_c, we can extract the number of times each software is used.

3) Aging Models with Uncertainty: Aging and degradation models are continuously under study and researcher develop more accurate models everyday. We generalize our HSCM method to achieve minimum error in software metering in the presence of uncertainty in aging models. Let's assume that the gate usage time t is a function of ΔV_{th}; $t = \Theta(\Delta V_{th})$ with some uncertainty υ. υ is a random variable which can possess different probability distributions. We assume υ has a normal distribution. Therefore, usage time for gate i can be expressed as:

$$\begin{aligned} t_i &= \Theta(\Delta V_{th_i}) + \upsilon_i \\ &= \Theta(\frac{d_i - d_0}{d_0}\frac{V_{gs} - V_{th}}{\alpha}) + \upsilon_i \\ &= \Theta_d(d_i) + \upsilon_i \end{aligned} \tag{14}$$

Θ_d is the composition of delay-threshold voltage and threshold voltage-aging functions. As we discussed in Section IV-D2, gate usage time is in fact the total running time of software that cause stress on that gate:

$$t_i = \sum_{j=1}^{j=r_i} \tau_{i_j} \tag{15}$$

the above sum is takes over all software in φ_i. Equations 14 and 14 lead to the following set of linear equations with gaussian noise:

$$t_i = b_i^T \bar{\tau} + \upsilon_i, \forall 1 \leq i \leq k \tag{16}$$

where b_i is the vector which represent which software contributes to t_i:

$$\begin{aligned} b_{ij} &= 1, \forall S_j \in \varphi_i \\ &= 0, otherwise \end{aligned} \tag{17}$$

Equation 17 is similar to 5 and can be solved in a similar fashion. Uncertainty and imperfections in aging models, may possess different properties and probability distributions. Therefore different uncertainty models can be incorporated in this formulation and be solved accordingly. The solution to set of equations in 16 is the running times of software which completes our HSCM method. At this stage, we use remote activation scheme that aims to protect integrated circuits (IC) and intellectual property (IP) [3].

V. Simulation Results

In this section, we simulate our HSCM method to prove the concept of hardware aging-based software metering, especially under the NBTI effect. We use an 8×8 butterfly aging circuit (which accepts 16-bit input vectors). In the simulation testbenches, we assume $k = 30$ different software applications may be run on the target device. At each point of time, we select each software randomly and assign a random execution time in the range of $[1000 - 5000]$ seconds. The device is assumed to operate for the total amount of 200000 seconds before the HSCM is performed. Once run times and execution ordering were determined, we find the subset of gates in the aging circuit that will be under stress for that particular software, and, using equation 3, the corresponding threshold voltages are shifted. In order to integrate model imperfections, we added a normal noise to our calculations; for path delay measurement, the standard deviation of the error is increased from 1 to 8 times of the initial delay of a gate. As for the model imperfections, we added a gaussian error with zero mean and standard deviation equal to 0.001 volts.

Once the aging circuit was artificially under stress, we measured the delay of m paths in the circuit with an injected random gaussian error. We then followed our HSCM steps. Simulation results are illustrated in Figures 3, 4 and 5.

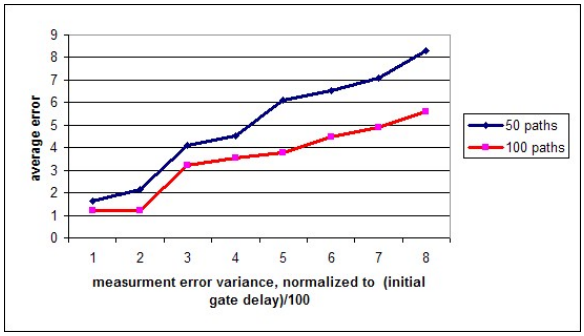

Fig. 3. Average software metering error for different delay measurement errors. Increase in the number of path delays in HSCM methos decreases the average error.

In Figure 3 we applied our HSCM method for two different scenarios; in the first run, we used the delay of 50 paths and in the second run we used the delay of 100 paths. The x-axis is the standard deviation of the measurement error which has a normal distribution. The std is a multiple of the initial delay of a NAND gate divided by 100. The y-axis, represent the average software usage error. As you can see, the worst

average error in the presence of maximum measurement error is around 8% when using only 50 paths and has been reduced to less that 6% by measuring the delay of 100 gates.

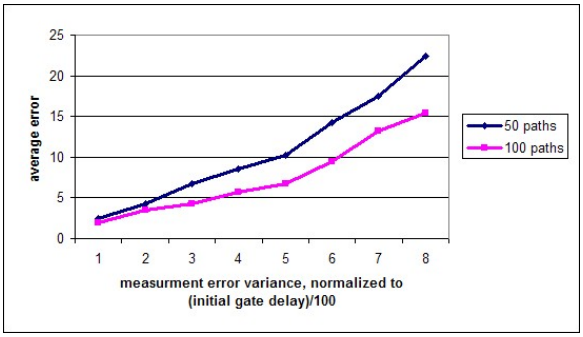

Fig. 4. This graph shoes that when software are run for a long time, ΔV_{th} converges to a constant value and aging looses its dynamic nature and therefore average error in software metering increases.

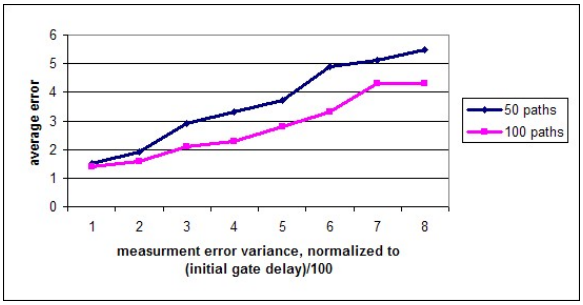

Fig. 5. When we scale up the aging circuit and feed the signature vectors 1/5 of the normal time, the accuracy increases again

Furthermore, we repeated the same procedure but this time, increased the total runtime of the software by a factor of 5. As we anticipated, the software metering error increased exponentially (4). The reason being is that, aging under NBTI is a concave function which informally converges to an almost constant amount. In other words, if you use the gates in a circuit for a long time, all gates will age too much which will fall into non-dynamic region of the curve in Figure 1. By non-dynamic, we mean the almost flat region of the curve which is less sensitive to usage time. In order to overcome this problem, we can use the following technique: Instead of feeding the signature vectors to the aging circuit for the whole amount of time the software is running, we can feed the vector for percentage of run time. This will reduce the aging factor and enables us to stay in the dynamic region (where the ΔV_{th} vs. stress time curve has larger derivative.

Figure 5 is the simulation results for the latter scenario where software execution time was an average 5 times larger than the first testbench. We used a Benes network (twice the aging circuit in the first scenario) In this case, each signature vector was fed to the aging circuitry for $\frac{1}{10}$ of the execution time. It is observed that average error rate is bounded by 6% and 5% percent for 50 and 100 path delay analysis which is even slightly better than the first scenario.

VI. Conclusion and Future work

In this paper, we addressed the problem of software/hardware metering through usage of hardware aging. In our approach we target the question of how we can measure the amount of time particular licensed software (LS) is used by designing an aging circuitry and exposing it to the unique inputs associated with each LS. If a particular LS is used longer than specified, it automatically disables itself.

Our novel HSCM technique uses a multi-stage optimization problem of computing the delays of gates, their aging degradation factors, and finally software metering using convex programming. The experimental results show not just viability of the technique but also surprisingly high accuracy in the presence of measurements noise and imperfect aging models. We did reach of an accuracy with less that 5% of average software metering error. HSCM can be used for many other businesses and engineering applications in e-commerce such as power minimization, software evaluation, and processor design.

There are a potentially huge number of variations and applications originating from our HSCM technique. In the future, we will study the utilization of AC stress to reverse the NBTI effect to some extent and therefore extend the usage time of our aging circuit and increase the number of software we can track. Further more, we believe more research on aging circuitry and signature vectors can lead us to architectures and input vectors which can increase the accuracy and usability of hardware aging based metering. Also, other hardware aging techniques will be explored, such as electromigration and interconnect aging, which will lead to novel security and IP management techniques in e-commerce.

References

[1] http://w3.bsa.org/globalstudy/.

[2] http://www.certicom.com/index.php?action=sol silicon/.

[3] Yousra Alkabani, Farinaz Koushanfar, and Miodrag Potkonjak. Remote activation of ics for piracy prevention and digital right management. In *ICCAD '07*, pages 674–677, Piscataway, NJ, USA, 2007. IEEE Press.

[4] H. Kufluoglu M. A. Alam B. C. Paul, K. Kang and K. Roy. Impact of nbti on temporal performance degradation of digital circuits. *IEEE Electron Device Letters*, 26(8):560–562, 2005.

[5] Matthew K. Franklin and Dahlia Malkhi. Auditable metering with lightweight security. In *FC '97*, pages 151–160, 1997.

[6] M. A. Alam K. Kang, H. Kufluoglu and K. Roy. Effcient transistor-level sizing technique under temporal performance degradation due to nbti. In *ICCD06*, 2006.

[7] Sanjay V. Kumar, Chris H. Kim, and Sachin S. Sapatnekar. An analytical model for negative bias temperature instability. In *ICCAD '06*, pages 493–496, New York, NY, USA, 2006. ACM.

[8] Moni Naor and Benny Pinkas. Secure accounting and auditing on the web. In *WWW7*, pages 541–550, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[9] James Pitkow. In search of reliable usage data on the www. In *Selected papers from the sixth international conference on World Wide Web*, pages 1343–1355, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[10] Jr. Robert Dixon Raymond Findley. Dual smart card access control electronic data storage and retrieval system and methods. In *US Patent, number 5629508*, 1999.

[11] B. Schneier and J. Kelsey. A peer-to-peer software metering system. In *in The Second USENIX Workshop on E-Commerce*, pages 279–286, 1996.

[12] Dieter K. Schroder. Negative bias temperature instability: What do we understand? *Microelectron. Eng.*, 47(6):841–852, 2007.