IMPLEMENTATION OF ALGORITHM FOR ANALYSIS OF MULTI TRAP RANDOM TELEGRAPH NOISE IN MOSFETs

A Dissertation Submitted to the

Department of Electrical Engineering of IIT Madras,

In partial fulfilment of the requirements for the degree of

M.Tech in Electrical Engineering

Submitted by

S Lakshmi Tirupathi Kumari

Roll No. EE19M052

Supervisor:

Dr. Deleep R Nair

Department of Electrical Engineering

Indian Institute of Technology, Madras

Department of Electrical Engineering

Indian Institute of Technology, Madras

Chennai 600036, Tamil Nadu, India

June 2021.

Acknowledgement

The present project work is submitted in fulfilment of the requirements for the degree of Master of Technology of Indian Institute of Technology, Madras (IITM). I express my deepest gratitude to my supervisor Dr. Deleep R Nair of Indian Institute of Technology, Madras for his inestimable support, encouragement, profound knowledge, largely helpful conversations and also for providing me a systematic way for the completion of my project works. I am also extremely grateful to the Pavan Ch, PhD student of IITM, for his continuous help and support throughout the project.

IITM

June 2021

(S Lakshmi Tirupathi Kumari)

Department of Electrical Engineering

Indian Institute of Technology, Madras.

ABSTRACT

Reliability of MOSFETs has become a major concern in scaled technologies. The potential of innovative technologies and processes can be realized only if the reliability constraints are in tolerable limits. Some of the important reliability aspects in scaled MOSFETs are bias and temperature instability (BTI), random telegraph noise (RTN), stress induced leakage currents (SILC), hot carrier degradation (HCD). One of these aspects, namely random telegraph noise has attracted attention of researchers as it poses a reliability challenge for scaled devices. It is defined as fluctuation in source-drain current of a MOSFET due to trapping and detrapping of charge carrier from the channel into a gate oxide trap. This results in fluctuation in threshold voltage of the device (ΔV_T). A single trap in the gate insulator can cause fluctuation involving two discrete levels in drain current. If n-traps become active, it can result in $2^{n-1}+1$ to 2^n discrete levels in the drain current. RTN can impact digital circuits, SRAMs and Flash memory cells.

Multi trap RTN is also referred to as complex RTN and it can cause large shifts in threshold voltage of the device. Further, the analysis also gets complicated for multi trap RTN. One of the important steps in multi trap RTN analysis is to decompose the multi-level RTN as a superposition of two level RTNs. Some of the algorithms used are based on bayesian inference, hidden markov models. These are probability based algorithms which are computationally intensive and time consuming. So, non-probabilistic algorithms are more preferred.

One such algorithm is implemented in this thesis which is based on a patent. This thesis is focused on implementation of an algorithm for analysis of multi-trap RTN. The implementation of the algorithm is divided into two stages-histogram based analysis and transition based analysis. The multi trap RTN data is given to first stage whose output will be the number of traps, RTN amplitude of each trap, time constants ratios of each trap. After this, the second stage processing involves identification of transitions between traps and computing individual time constant values of each trap.

The input to the algorithm is a multi-trap RTN measured on a pFET. The aim of the algorithm implemented in this work is to extract the individual trap components-its RTN amplitude and time constants. As part of the project, first stage analysis is implemented in MATLAB and results were shown.

CONTENTS

CHAPTER 1

INTRODUCTION TO RTN

1.1 Introduction to Random Telegraph Noise (RTN) in MOSFET

Random Telegraph Noise (RTN) is an important reliability aspect in scaled MOSFETs which has attracted the attention of researchers. It was first observed in drain current of MOSFET in the year 1984 and the physical mechanism was been identified as trapping and detrapping events occurring between the charge carriers in the channel and gate oxide traps or interface traps[1,2].

1.1.1 Physical mechanism of RTN

Random Telegraph Noise (RTN) is caused by random trapping and detrapping of a single charge carrier from channel into gate oxide traps. This manifests as a two level fluctuation in drain current of MOSFET. A typical diagram depicting the RTN is shown below in Fig.1. RTN has become a reliability concern for digital circuits[3], SRAM[4,5], Flash memories[6].

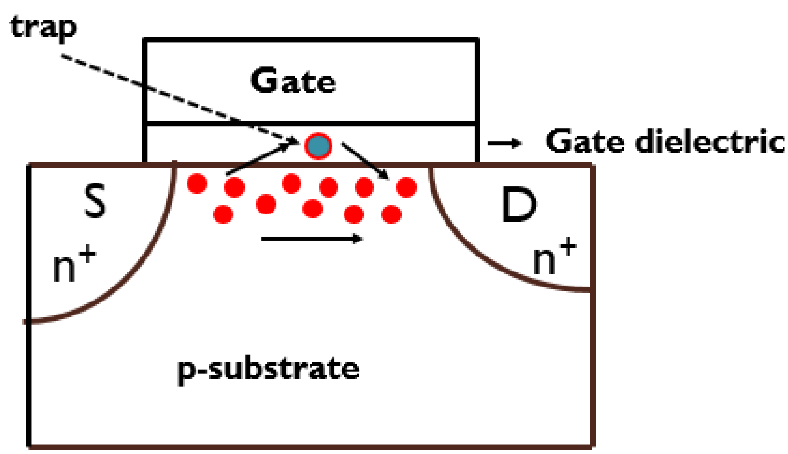

Fig 1.1 : Schematic representing the capture and emission of an electron from the channel into gate oxide trap

1.1.2 Time domain and frequency domain properties of RTN

Any noise can be characterized in time domain as well as frequency domain. A typical RTN waveform is shown below. In time domain, it is characterized by three primary parameters-namely capture time, emission time and RTN amplitude. The capture time is defined as average time to capture a charge carrier (electron or hole) into the gate oxide trap, emission time is the average time to emit the charge carrier into channel.

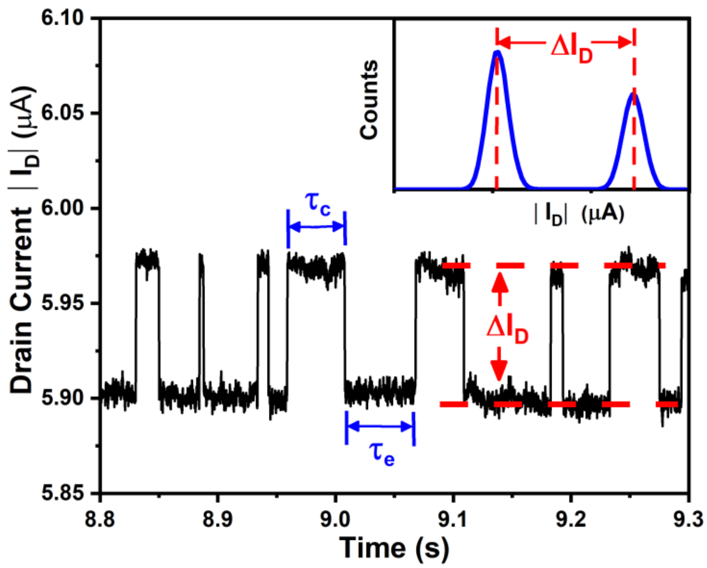

Fig 1.2: Typical RTN waveform denoting the capture and emission time constants. The figure in the inset shows the histogram of drain current RTN with two gaussian peaks(Courtesy: Pavan Ch)

RTN is a random process and the time durations of trap being empty and full (capture and emission durations) follow exponential distribution [1]. The capture and emission times are the mean values extracted by fitting the capture and emission durations to exponential distribution as illustrated in the figure below.

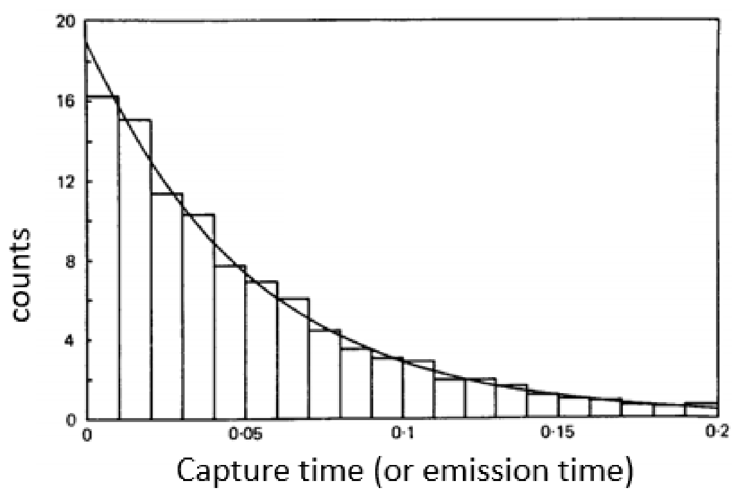

Fig 1.3: Schematic showing the exponential distribution of capture and emission durations[1]

RTN amplitude is defined as the difference of the discrete levels observed in RTN. It is computed as difference of peaks in the histogram as shown in figure below. Further, the RTN amplitude can be transformed into threshold voltage fluctuation (ΔV_T) given as

$$\Delta V_T = \frac{\Delta I_D}{G_m}$$

where ΔI_D is the difference in the discrete levels in the histogram, Gm is the transconductance of the device.

The impact of RTN for a given technology is expressed in terms of RTN induced ΔV_T. Specifically statistical measurements are performed and statistical averages of ΔV_T is obtained.

In frequency domain, the RTN is characterized by a lorentzian spectrum as shown below. The important parameters of PSD denoted as S(f) are S(0) which indicates the constant plateau at lower frequencies, f_o represents the corner frequency. The PSD falls with $1/f^2$ for frequencies greater than f_o.

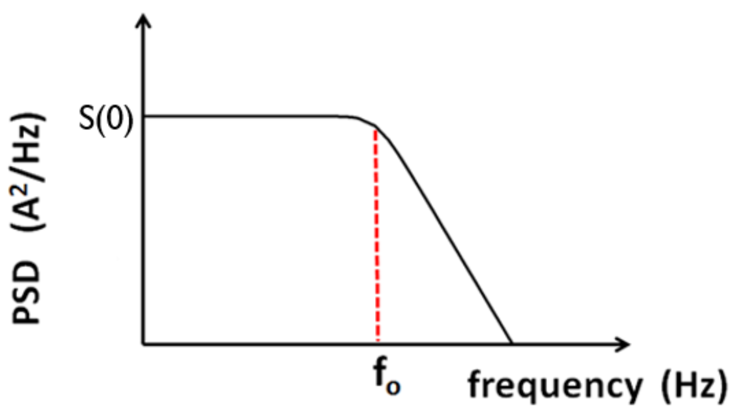

Fig 1.4: Schematic showing the Lorentzian spectrum for RTN

The PSD is given as

$$\boldsymbol{S}(\boldsymbol{f}) = \frac{\boldsymbol{S}(0)}{1 + \left(\frac{\boldsymbol{f}}{\boldsymbol{f_o}}\right)^2}$$

Where the corner frequency is related to time constants as

$$\boldsymbol{f_o} = \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{\pi}}\left(\frac{\mathbf{1}}{\boldsymbol{\tau_c}} + \frac{\mathbf{1}}{\boldsymbol{\tau_e}}\right)$$

The above equation also represents the relation between time domain and frequency domain parameters of RTN.

1.1.3 Relation between flicker noise and RTN

Low Frequency Noise(1/f noise) can be expressed as a summation of Lorentzians with a specific distribution of time constants that is uniform on a log scale.[7] RTN dominates for small areas devices, while LFN for larger area devices.

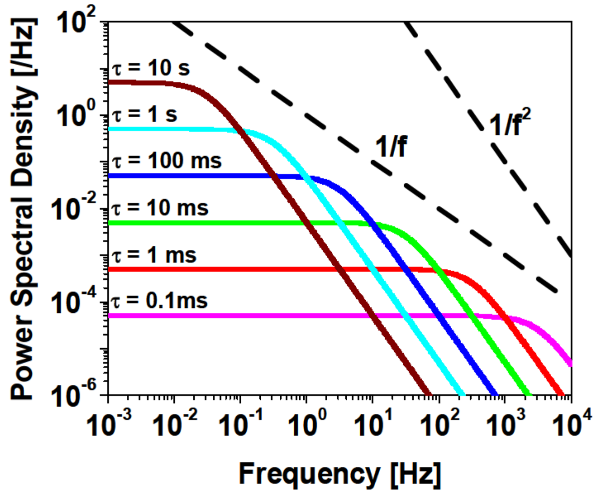

Fig 1.4: Superposition of lorentzians caused due to individual trap results in 1/f spectrum[8].

1.1.4 Impact of RTN

RTN has become a reliability challenge at scaled devices. A brief outline of RTN impact is given below.

1) The RTN induced threshold voltage ΔV_T increases at lower gate lengths, which indicates that RTN can pose a serious challenge at lower nodes [9]

2) RTN can induce performance fluctuations in CMOS circuits[3,10]

3) RTN affects the SRAM stability and also minimum operating voltage V_{min} [4,11]

4) RTN can cause fluctuations in read current in flash memory cells [12]

1.2 Multi Trap RTN

1.2.1 Introduction to multi trap RTN

A single trap in the gate oxide produces two discrete levels in the drain current of a MOSFET. If more than one trap is active in trapping and detrapping of inversion layer carriers, then it leads to more than two levels in the RTN measurement. In general, if there are n-traps it can cause $2^{n-1}+1$ to 2^n levels. Multi trap RTN also called as complex RTN. It can cause large ΔV_T values of the device. Some example of RTN with two traps are shown below. It can be observed from the figure that two traps resulted in four discrete levels in drain current of MOSFET.

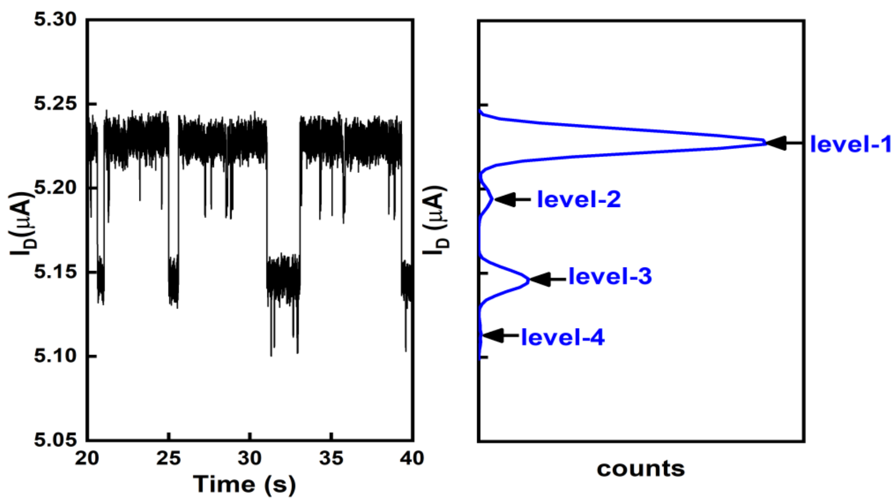

Fig 1.5 : Example of a four level RTN due to two traps measured on pMOSFET (Courtesy : Pavan Ch)

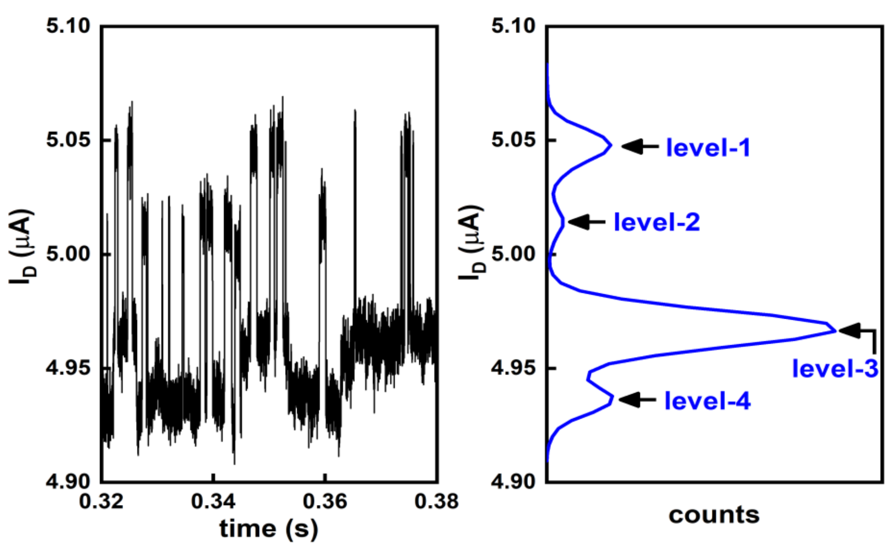

Fig 1.6 Example of a four level RTN due to two traps measured on pMOSFET (Courtesy : Pavan Ch)

1.2.2 Algorithms multi-trap RTN data

For multi trap RTN, the first and important step is to decompose the multi trap RTN into superposition of two-level RTNs. An example of such a decomposition is shown below[13]. It can be observed that the discrete levels (shown in red color) have been identified from the measured multi trap RTN (shown in black color). Further, it is decomposed into superposition of two traps, each of the trap being a two level RTN.

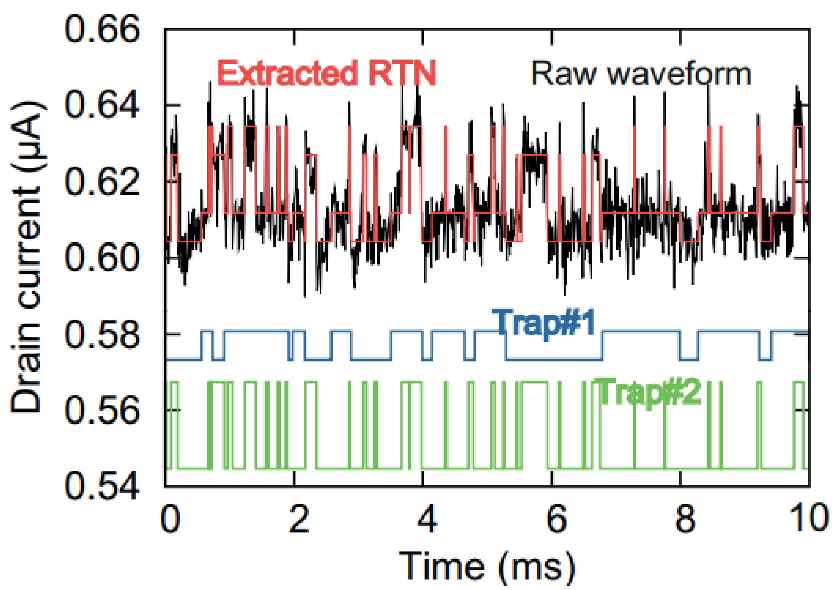

Fig 1.7 Decomposition of 4-level RTN into superposition of two-level RTNs[13]

The algorithm for analysis of multi trap RTN can be categorized into two groups-probability based and non-probabilistic algorithms.

Some of the examples of probability based algorithms are hidden markov models[14,15], factorial hidden markov models[16,17], Bayesian inference[18]. These are computationally intensive, time consuming.

The other group of algorithms are mostly based on dynamic thresholding techniques [19]. These are not time consuming and can run in reasonable time even with large number of data points in the measurement. Hence, these are preferred.[20]

CHAPTER 2

DESCRIPTION OF ALGORITHM FOR MULTI TRAP RTN DATA ANALYSIS

2.1 Introduction to the algorithm

This algorithm is implemented in two stages. A brief overview of the two stages is shown in the figure 2.1. The input and output of each of the stages is also depicted.

The first stage involves histogram based analysis. The measured RTN data is taken and histogram is obtained. Based on the number of peaks in the histogram, number of traps are identified. The number of traps, RTN amplitude of each trap and ratio of time constants of each trap are obtained as outputs of first stage.

Further, the second stage is referred to as transition based analysis. It involves assigning each data point in the RTN time series is assigned to the stable states obtained in the first stage. Further, the transitions for each of the traps are identified and time constants are computed for each of the traps.

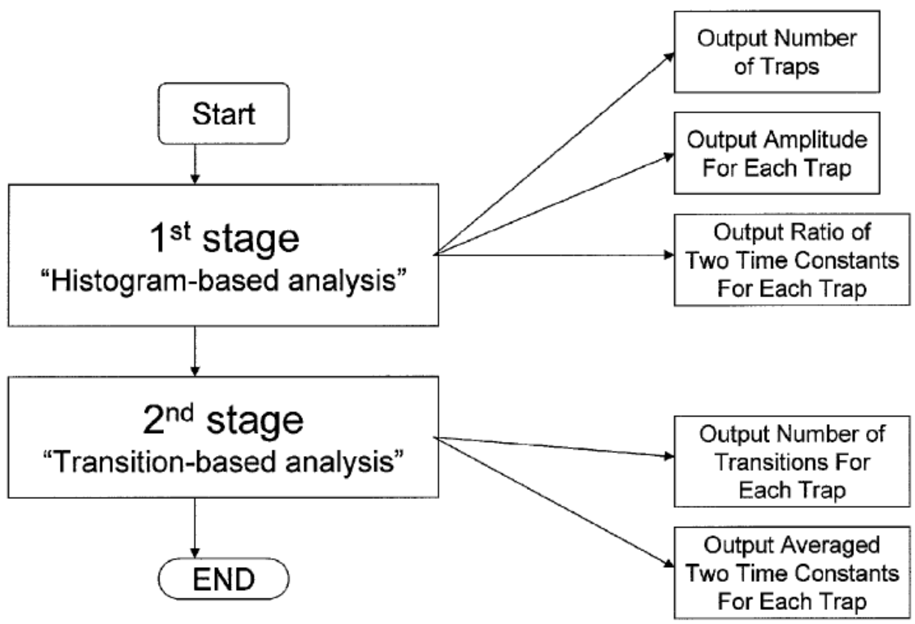

Fig 2.1 Schematic of two stage analysis of algorithm[20]

2.2 First stage analysis

In first stage analysis of the algorithm, histogram is obtained for RTN data time series. The number of peaks are counted in the histogram to identify number of traps. If there are n-traps it can result in $2^{n-1}+1$ to 2^n levels (peaks) in the histogram. The first stage analysis involves three steps, each step can be explained using a flowchart.

2.2.1 Extraction of characteristic physical constants

The process of implementation of first stage analysis is depicted in the flowchart as shown below in figure 2.2

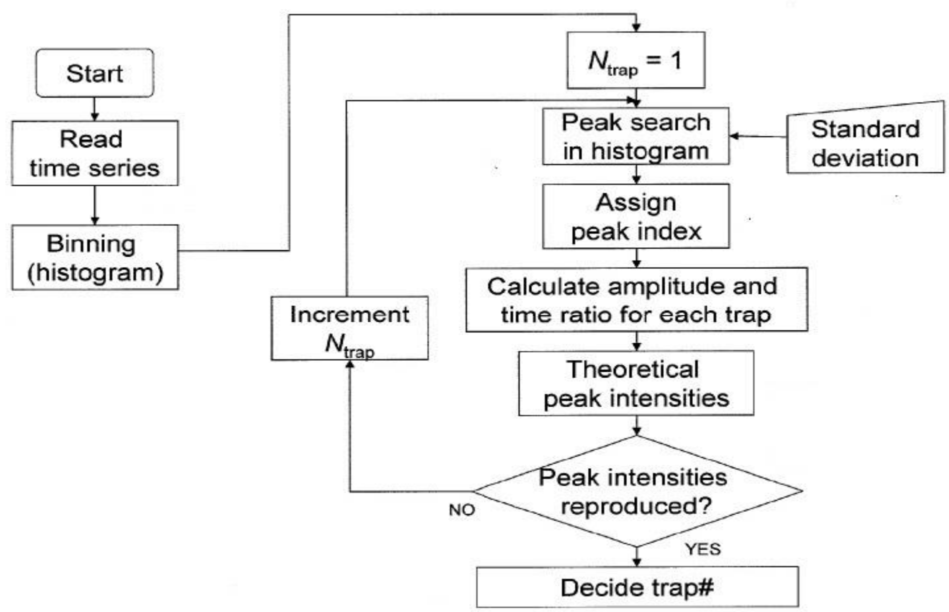

Fig 2.2 Flowchart for extraction of physical constants of RTN[20]

The measured time series RTN data is taken and binning is performed to obtain histogram. A counter representing number of traps in initiated with an initial value of 1. (N_{trap}=1). Then, the algorithm proceeds to search peaks of the histogram using standard deviation values. Number of peaks in histogram gives the information about number of traps. For example, if we have 2 states, it indicates presence of only one trap present. Similarly, if there are 4 states and 8 states, it corresponds to two and three traps respectively.

In the next step, each peak in the histogram is assigned a binary index value. Further, amplitude and a time constant ratio is calculated for each of the traps. The theoretical intensities are then determined for each of the peaks.

In the next step, the algorithm verifies whether the theoretical intensities are determined for all peaks. Further, reproduced theoretical peak intensities are verified. If not, then the number of traps determined is incremented by 1, and the steps from peak search in histogram to checking of peak intensities are repeated. If the theoretical peak intensities have been reproduced correclty, then the method determines the number of traps present, the method of the first stage determines a number of traps based on a correspondence between the calculated amplitude and time constant ratio for each trap and the theoretical intensity for each peak associated with each trap.

2.2.2 Recursive method of assigning peak index values

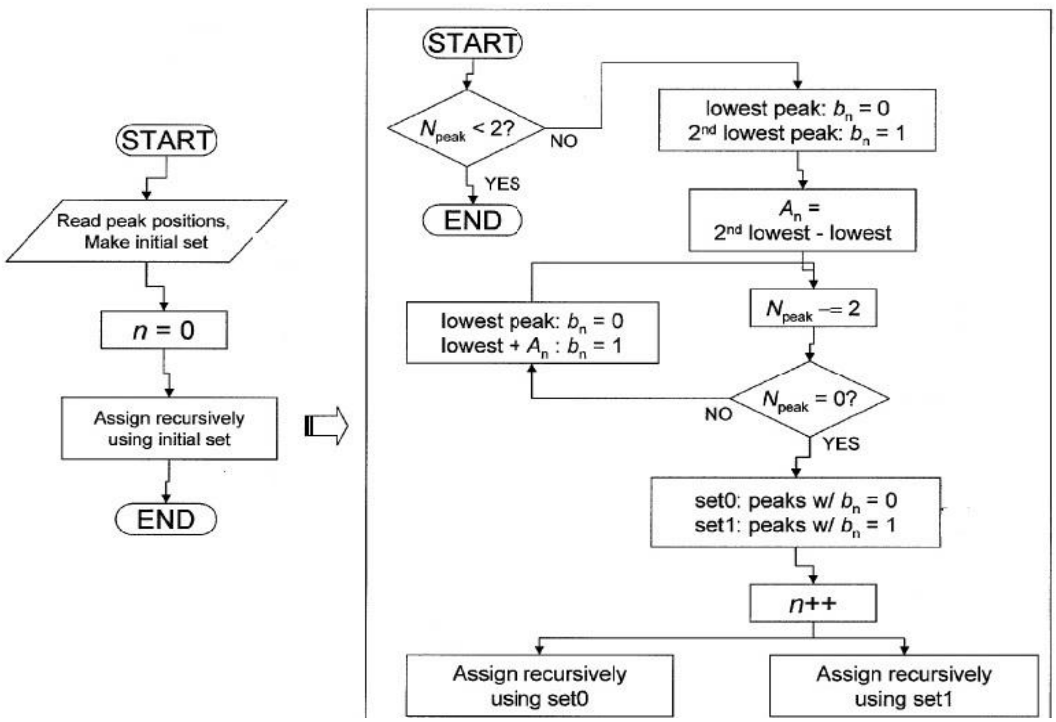

Fig 2.3 flowchart for recursively assignment of peak indexes [20]

The step of assigning a peak index value can be performed according to various techniques. For example, Fig2.5 is a flowchart of a recursive method of assigning peak index values in the first stage.

Referring to fig2.5, the method of recursively assigning a peak index value includes obtaining the peak positions from the time series and constructing an initial set of peak positions. The trap number count is set to 0. Then, peak indexes are assigned recursively using the next procedure. For example, the recursive assignment of peak indexes determining if the number of peaks is less than two, if so, then this step has been completed. If not, then this step includes assigning a lowest peak index value to one of the states in which all of the states is comprised of a first state value indicating the empty trap condition (for example, 0) and a second state value indicating the capture condition (for example, 1).

The transition from the first state value to the second state value represents capture of an electron by an associated trap, and a transition from the second state value to the first state value represents release of an electron from the associated trap. The drain current of the semiconductor device changes between at least two stable states in response to each transition.

The method continues by assigning the first state value indicating the empty condition to a lowest peak other than the next lowest peaks associated with next step. An offset value (An) is calculated as the difference in position between the lowest and the next lowest peaks in the next step. The number of peaks is then decremented by 2, and the method checks to see if there are any remaining peaks to be assigned. If there are more peaks to be assigned, then the method determines a peak which is at the lowest position in the remaining peaks, and the nearest in position in which the peak value is the lowest peak plus the offset value An determined. In next step the lowest peak is given an empty state with respect to the trap number of interest. The other peak which is positioned at An from the lowest peak is given an occupied state similarly. The steps from checking of Npeak value to checking of if there are any remaining peaks to assign, are repeated for all peaks in the histogram.

The method of recursive assignment continues by classifying all peaks in the histogram into one of a first set consisting of peaks assigned the first state value indicating the empty condition and as second set consisting of peaks assigned the second state value indicating the capture condition in next step. Then increase the n value indicating the next bit assignment of each peak. In further steps perform the recursive assignment for both first set and second set to assign further bit assignment.

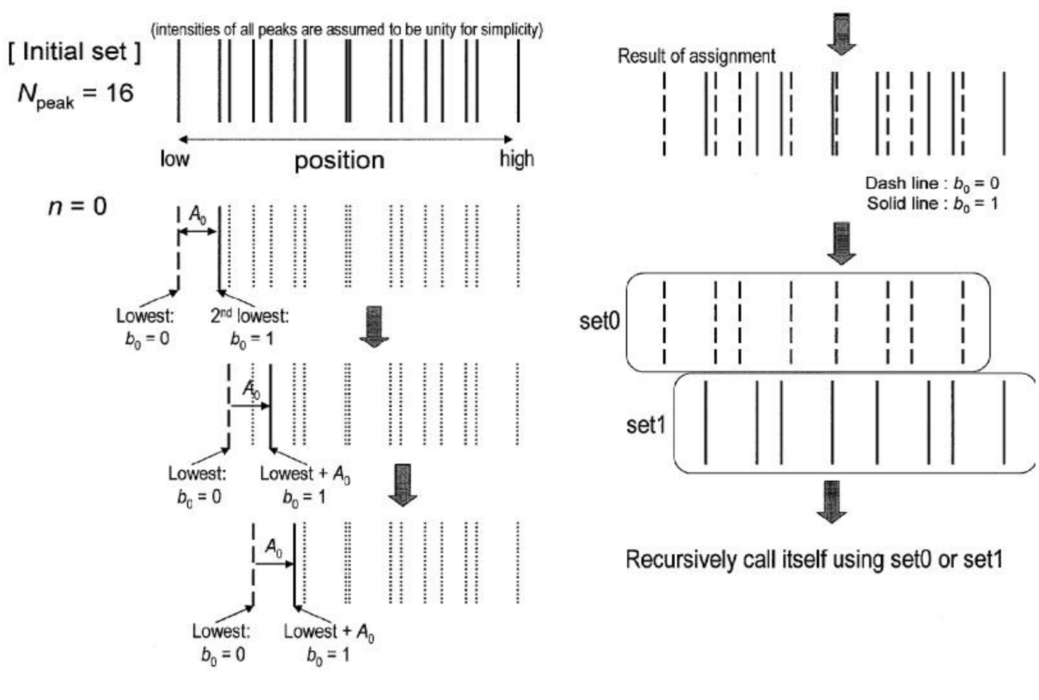

Fig 2.4 illustration of recursive assignment of peak indexes[20]

2.2.3 Calculation of amplitude and time constant ratio for individual traps

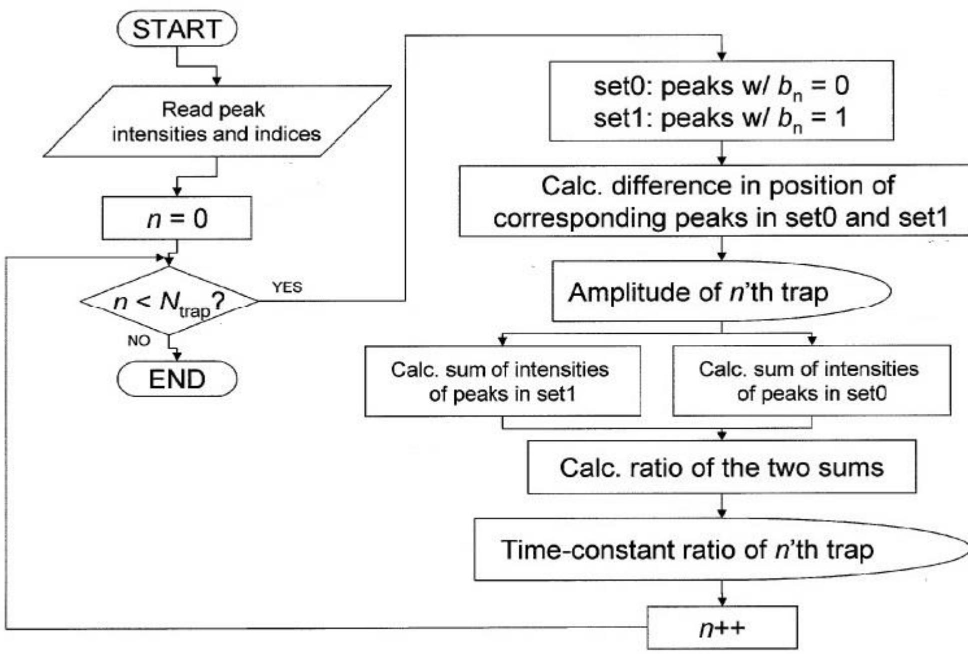

Fig 2.5 Flowchart for calculating amplitude and time constant for each trap[20]

The step of calculating an amplitude and a time constant ratio for each trap can be performed according to various techniques. Fig 2.7 is a flowchart of a method of calculating an amplitude and a time constant ratio for each trap in the first stage.

Referring to fig2.7, the method of calculating an amplitude and a time constant ratio of each trap can include obtaining or reading the peak intensities and indices at initial step, and setting an initial index value to 0 as taking n value as zero. Then at next step, it is determined if the index value is less than the number of traps. If not, then the result has been completed. If so, then the method further includes obtaining or reading the first set (consisting of peaks assigned the first state indicating the empty condition) and the second set (consisting of peaks assigned the first state value indicating the capture condition). Next, for each trap, the difference in position between corresponding peaks in the first set and the second set are calculated. Based on the difference, an amplitude is determined. Next, a first sum is calculated based on intensities of the peaks of the first set, and a second sum is calculated based on intensities of the peaks of the second set. Then, a ratio of the first and second sums is determined. Based on the ratio, time constant ratio is determined. the index value n is incremented by 1, and the method returns to checking of n value whether it is less than the number of traps at which the steps checking of index value through increment of n value are repeated.

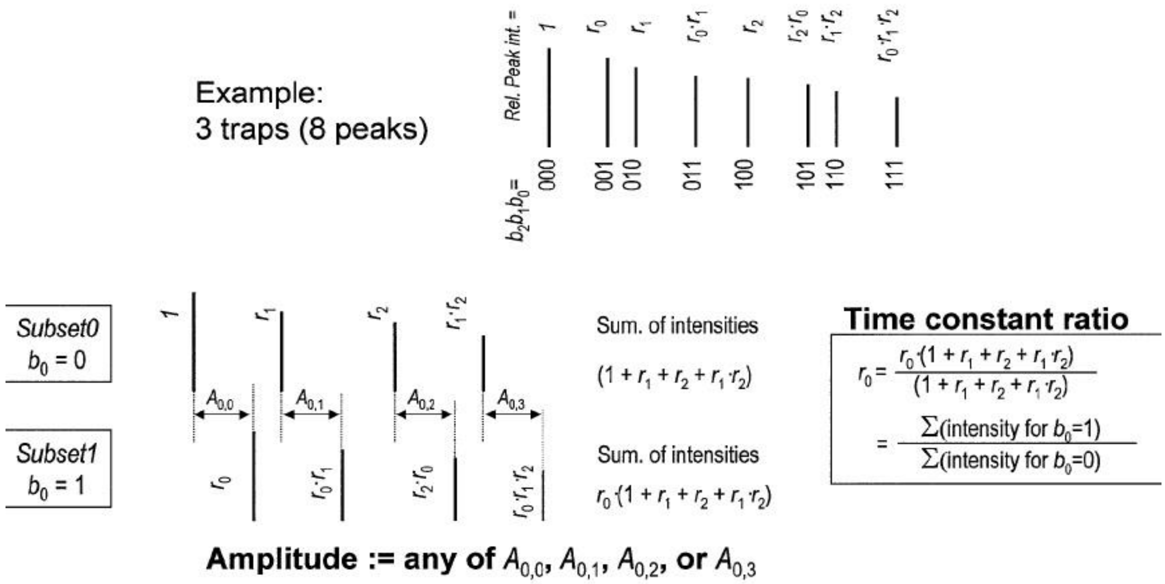

Fig 2.6 illustration of the calculating an amplitude and a time constant ratio for each trap[20]

2.3 Second stage analysis

2.3.1 Calculation of time constants through transition-based analysis

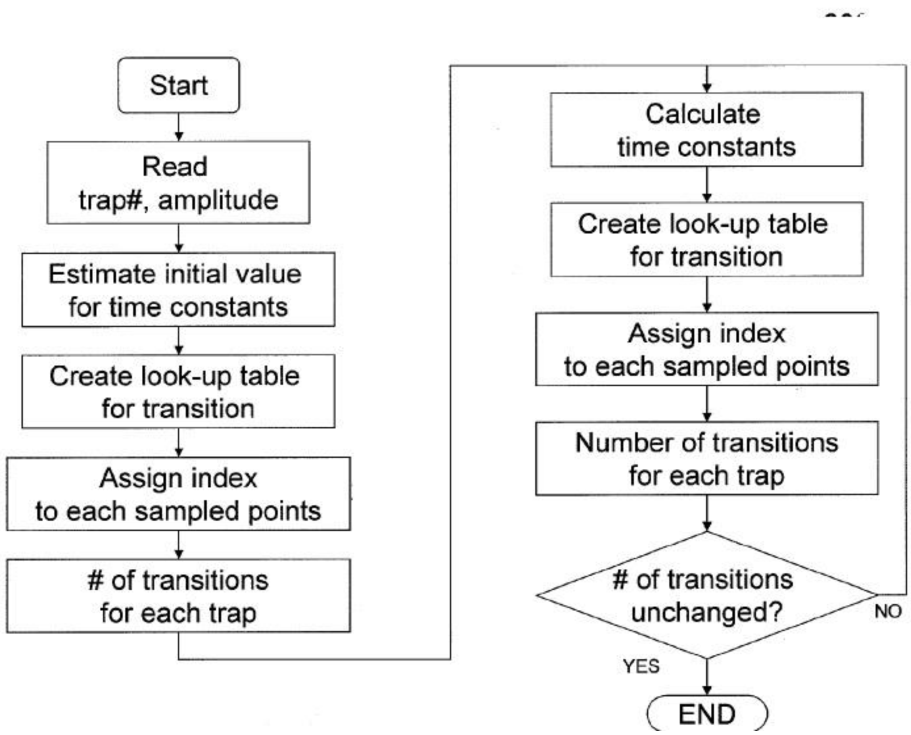

Fig 2.7 flowchart for calculating time constant through transition-based analysis

Referring to Fig2.9, a method can include reading or inputting the number of traps and their amplitudes from the first stage (refer to fig 2.4), and then estimating initial values for the corresponding time constants. Next, the method includes creating a look-up table defining a number of allowed next-state transitions from each state Then, an index value is assigned to each sampled point or value of the time series, and the number of transitions for each trap is determined. Following this determination, the time constants are calculated for each trap using the relationship between average holding time and transition rate, which is explained at Equation (1). Then, a revised look-up table defining the allowed next-state transitions from each state is computed, followed by revised index value assignment to each sampled point or value of the time series. Next, the number of transitions for each trap is determined again. Then in next step if the number of transitions determined for each trap is different from the previous determination, then steps from calculating time constant to checking of number of transition value are repeated recursively until the number of transitions determined is unchanged from the previous determination, in which case the method ends. In this way, the assignment of states is uniquely determined.

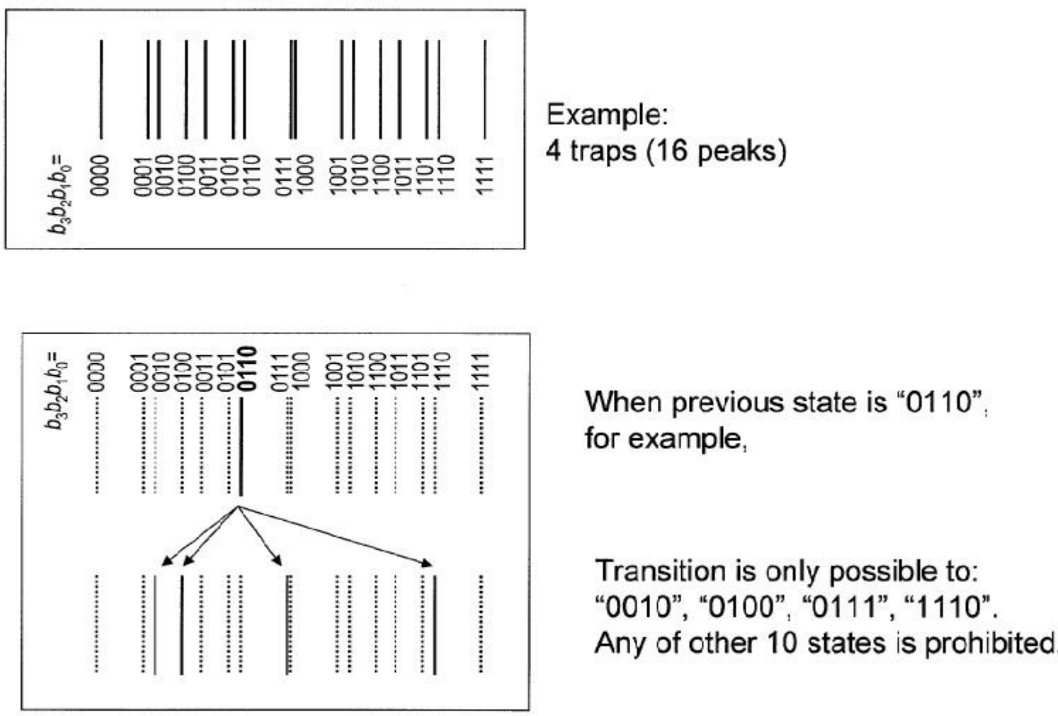

Fig2.8 illustrate method of inhibiting next-state transitions in creating a look-up table defining allowed state transitions[20]

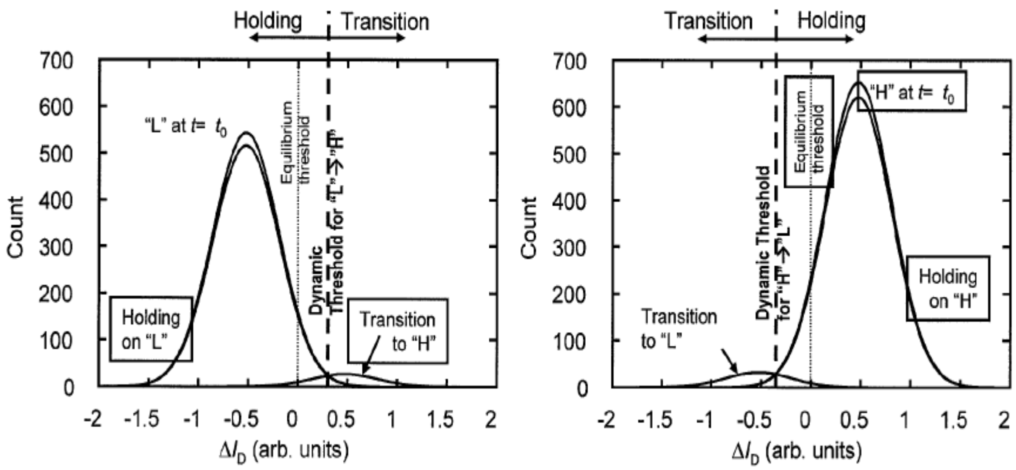

Fig 2.9 illustrate a method of determining transition-based assignments[20]

2.3.2 Method of creating a look-up table defining number of allowed next- state transitions from each state.

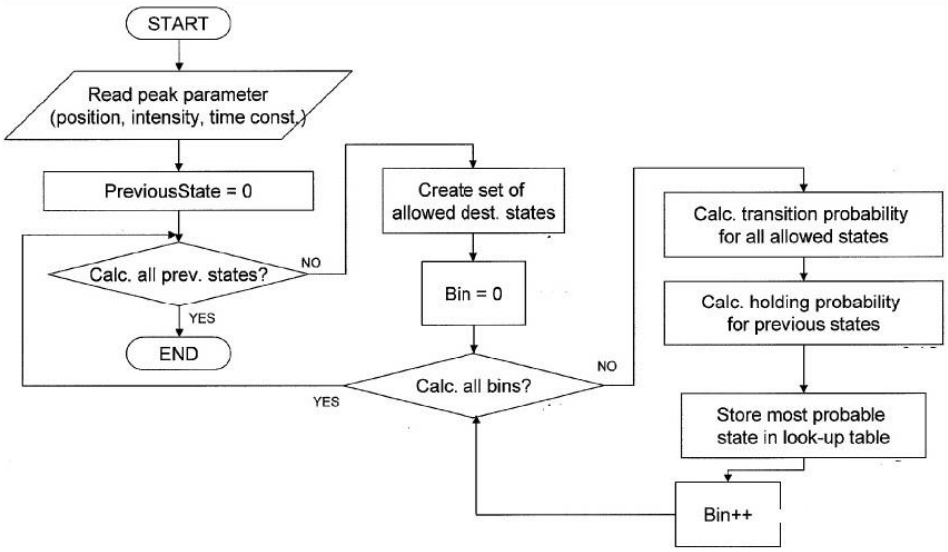

Fig 2.10 flowchart for Method of creating a look-up table defining number of allowed next-state transitions from each state[20]

Referring to Fig 2.12, the method of creating a look-up table defining a number of allowed next-state transitions from each state includes obtaining or reading peak parameters such as, for example, peak position, intensity, and associated time constants.

Initially, a previous state counter is set to 0. Then it is determined if the look-up table contents for all previous states have been calculated. If not, then the method further includes creating a set of allowed destination states as discussed here in above with respect to fig2.10. Next, the bin for the destination state being processed is set initially to 0. Then at next state, it is determined if all bins for the state being processed have been calculated. If so, then the method returns to checking of previous states transition. If not, then the method continues to next step at which the transition probability is calculated for all allowed states, and then to next step at which the holding probability is calculated for the holding state, as described, for example, with respect to figs2.11. Next, the most probable next state is stored in a computer –readable storage medium or memory, the bin counter is incremented by 1, and the method returns to state at which checking of bins calculation.

CHAPTER 3

PROGRAMMING TO CALCULATE RTN TIME CONSTANT

Origin Graphic tool to draw histogram and calculate standard deviation of measured data:

Matlab code for two trap:

```
clc;

clear all;

close all;

format long;

filename='C:\Users\laksh\Documents\MATLAB\50usec_Channel101.csv';

delimiter = ','; startRow = 1; endRow = inf;

Noise=importdata(filename, startRow, endRow);

Time=Noise(:,1);

Id=abs(Noise(:,2));

[binpos,ftshist]=hist(Id,80);

plot(ftshist,binpos);

List_peaks = 1E-06*[3.7357,3.8078,3.8783,0];

% List_peaks = 1E-06*[2.5 2.62 2.8 2.9 2.98 3.4 3.52 3.6 3.72 3.79 3.85 4.0 3.46 3.15 2.75 3.33];

% List_peaks = 1E-06*[2.5 2.62 2.8 2.9 2.98 3.4 3.52 3.6];

Npeak=length(List_peaks);

Nvalue=Npeak;

Nbit=log2(Npeak);

res=zeros((Nbit+1),Npeak);

% set0=zeros(1,Npeak/2);

% set1=zeros(1,Npeak/2);

for i=1:Npeak
```

```
res(1,i)=List_peaks(1,i);
end
result=recursive(Nvalue);
result1=myrecursive(List_peaks);
N1=length(result1);
for k=1:(N1/2)
   set0(1,k)=result1(1,k);
   set1(1,k)=result1(1,(k+(N1/2)));
end
for i=1:Npeak
   for j=1:Npeak
      if res(1,i)==result1(1,j);
         res(2,i)=result1(2,j);
      end
   end
end
result2=myrecursive(set0);
result3=myrecursive(set1);
result1=[result2,result3];
for i=1:Npeak
   for j=1:Npeak
      if res(1,i)==result1(1,j);
         res(3,i)=result1(2,j);
      end
   end
end
```

```
% N2=length(result2);
% for k=1:(N2/2)
%     set2(1,k)=result2(1,k);
%     set3(1,k)=result2(1,(k+(N2/2)));
%     set4(1,k)=result3(1,k);
%     set5(1,k)=result3(1,(k+(N2/2)));
% end
% result2=myrecursive(set2);
% result3=myrecursive(set3);
% result4=myrecursive(set4);
% result5=myrecursive(set5);
% result6=[result2,result3];
% result7=[result4,result5];
% result1=[result6,result7];
% for i=1:Npeak
%     for j=1:Npeak
%         if res(1,i)==result1(1,j);
%             res(4,i)=result1(2,j);
%         end
%     end
% end
% N3=length(result5);
% for k=1:(N3/2)
%     set6(1,k)=result2(1,k);
%     set7(1,k)=result2(1,(k+(N3/2)));
%     set8(1,k)=result3(1,k);
%     set9(1,k)=result3(1,(k+(N3/2)));
```

```
%     set10(1,k)=result4(1,k);
%     set11(1,k)=result4(1,(k+(N3/2)));
%     set12(1,k)=result5(1,k);
%     set13(1,k)=result5(1,(k+(N3/2)));
% end
% result2=myrecursive(set6);
% result3=myrecursive(set7);
% result4=myrecursive(set8);
% result5=myrecursive(set9);
% result6=myrecursive(set10);
% result7=myrecursive(set11);
% result8=myrecursive(set12);
% result9=myrecursive(set13);
% result10=[result2,result3];
% result11=[result4,result5];
% result12=[result6,result7];
% result13=[result8,result9];
% result14=[result10,result11];
% result15=[result12,result13];
% result1=[result14,result15];
% for i=1:Npeak
%     for j=1:Npeak
%         if res(1,i)==result1(1,j);
%             res(5,i)=result1(2,j);
%         end
%     end
% end
```

```
%//code for caluculating amplitude and time constant ratio of each trap//

dummy0=0;
  dummy1=0;
  new_set0=zeros(Nbit,(Nvalue/2));
  new_set1=zeros(Nbit,(Nvalue/2));
  timeconstant_ratio_trap=zeros(1,Nbit);
  amplitude=zeros(Nbit,(Nvalue/2));
  j=1;l=1;
  for i=1:1:Nbit
  for k=1:1:Nvalue
     if res((i+1),k)==0
        dummy0=dummy0+res(1,k);
        new_set0(i,j)=res(1,k);
        j=j+1;
        if j==((Nvalue/2)+1)
           j=1;
        end
     else if res((i+1),k)==1
        dummy1=dummy1+res(1,k);
        new_set1(i,l)=res(1,k);
        l=l+1;
        if l==((Nvalue/2)+1)
           l=1;
        end
        end
```

```
    end


  end

    timeconstant_ratio_trap(1,i)=(dummy0/dummy1); %output ratio of two time constants for each trap

    amplitude=new_set0-new_set1;

  end

  for i=1:Nbit

    for k=1:(Nvalue/2)

      if amplitude(i,k)<0

        amplitude(i,k)=-amplitude(i,k); % output amplitude for each trap

      end

    end

  end
```

Output of the code:

1) **trap count:**

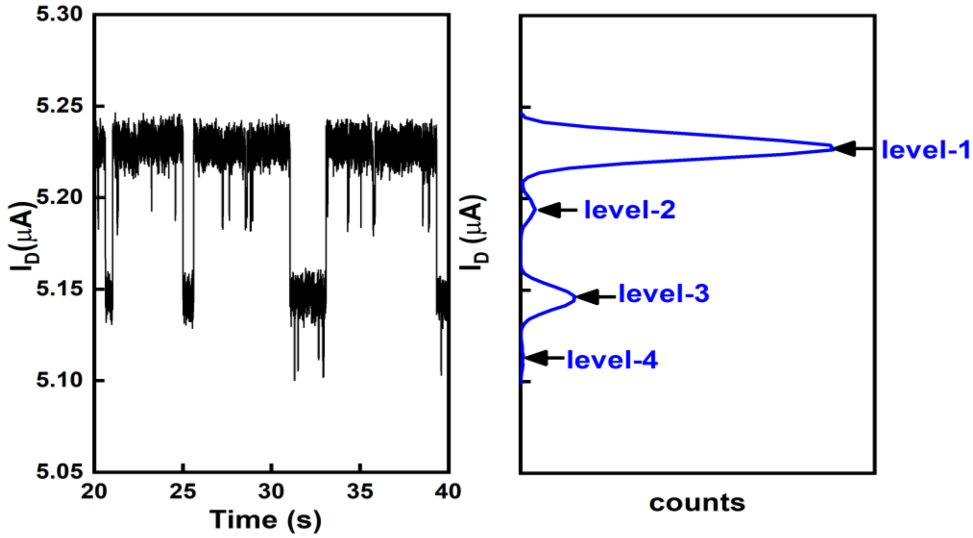

Fig: four-level RTN waveform and its histogram

2) **peak index assigning:**

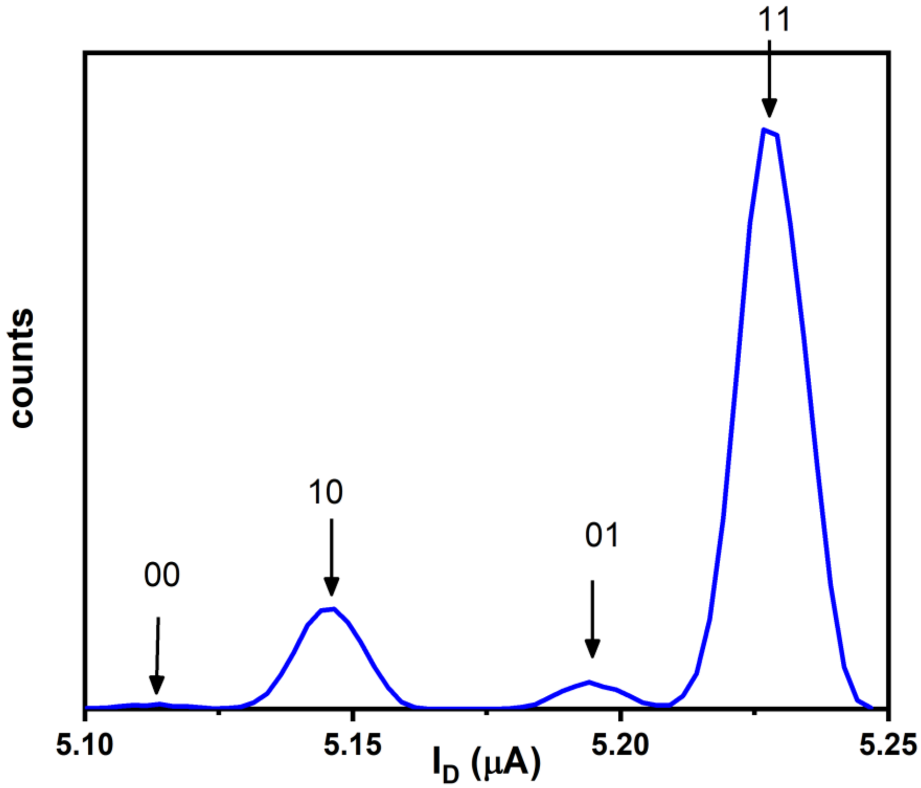

Fig :Assignment of peaks with binary indexes

Peak value (μA)	Binary value
5.11	00
5.149	10
5.198	01
5.23	11

Table 1: peak assignment

3) Amplitude and time constant ratio of each trap from 4-level RTN

Trap	RTN amplitude	RTN amplitude	Time constant ratio
Trap-1	39 nA	32 nA	0.9932
Trap-2	88 nA	81 nA	0.9985

Matlab code for calculating number of traps and assigning peak indexing in first stage (it works for max four traps):

```
% For analysis purpose here we are taking 16 peak stages of data

clc;

clear all;

close all;

format long;

filename='C:\Users\laksh\Documents\MATLAB\50usec_Channel101.csv';

delimiter = ','; startRow = 1; endRow = inf;

Noise=importdata(filename, startRow, endRow);

Time=Noise(:,1);

Id=abs(Noise(:,2));

[binpos,ftshist]=hist(Id,80);

plot(ftshist,binpos);

initialset=1E-06*[3.7357,3.8078,3.8783,0]; %actual peak values calculated using origin tool

List_peaks = 1E-06*[2.5 2.62 2.8 2.9 2.98 3.4 3.52 3.6 3.72 3.79 3.85 4.0 3.46 3.15 2.75 3.33];

% List_peaks = 1E-06*[2.5 2.62 2.8 2.9 2.98 3.4 3.52 3.6];

Npeak=length(List_peaks);

Nvalue=Npeak;

Nbit=log2(Npeak);
```

```
res=zeros((Nbit+1),Npeak);
% set0=zeros(1,Npeak/2);
% set1=zeros(1,Npeak/2);
for i=1:Npeak
res(1,i)=List_peaks(1,i);
end
result=recursive(Nvalue);
result1=myrecursive(List_peaks);

   N1=length(result1);
for k=1:(N1/2)
   set0(1,k)=result1(1,k);
   set1(1,k)=result1(1,(k+(N1/2)));
end
for i=1:Npeak
   for j=1:Npeak
      if res(1,i)==result1(1,j);
         res(2,i)=result1(2,j);
      end
   end
end
result2=myrecursive(set0);
result3=myrecursive(set1);
result1=[result2,result3];
for i=1:Npeak
   for j=1:Npeak
      if res(1,i)==result1(1,j);
```

```
            res(3,i)=result1(2,j);
        end
    end
end

N2=length(result2);
for k=1:(N2/2)
    set2(1,k)=result2(1,k);
    set3(1,k)=result2(1,(k+(N2/2)));
    set4(1,k)=result3(1,k);
    set5(1,k)=result3(1,(k+(N2/2)));
end
result2=myrecursive(set2);
result3=myrecursive(set3);
result4=myrecursive(set4);
result5=myrecursive(set5);
result6=[result2,result3];
result7=[result4,result5];
result1=[result6,result7];
for i=1:Npeak
    for j=1:Npeak
        if res(1,i)==result1(1,j);
            res(4,i)=result1(2,j);
        end
    end
end
N3=length(result5);
```

```
for k=1:(N3/2)
    set6(1,k)=result2(1,k);
    set7(1,k)=result2(1,(k+(N3/2)));
    set8(1,k)=result3(1,k);
    set9(1,k)=result3(1,(k+(N3/2)));
    set10(1,k)=result4(1,k);
    set11(1,k)=result4(1,(k+(N3/2)));
    set12(1,k)=result5(1,k);
    set13(1,k)=result5(1,(k+(N3/2)));
end
result2=myrecursive(set6);
result3=myrecursive(set7);
result4=myrecursive(set8);
result5=myrecursive(set9);
result6=myrecursive(set10);
result7=myrecursive(set11);
result8=myrecursive(set12);
result9=myrecursive(set13);
result10=[result2,result3];
result11=[result4,result5];
result12=[result6,result7];
result13=[result8,result9];
result14=[result10,result11];
result15=[result12,result13];
result1=[result14,result15];
for i=1:Npeak
    for j=1:Npeak
```

```
        if res(1,i)==result1(1,j);
            res(5,i)=result1(2,j);
        end
    end
  End
```

Recursive function to calculate Ntrap value:

```
function output=recursive(N)
List_peaks = 1E-06*[2.5 2.62 2.8 2.9 2.98 3.4 3.52 3.6 3.72 3.79 3.85 4.0 3.46 3.15 2.75 3.33];
Nlength=length(List_peaks);
Nbit=log2(Nlength);
output=zeros((Nbit+1),N);
set0=zeros((Nbit+1),(N/2));
set1=zeros((Nbit+1),(N/2));
array=List_peaks;
j=1;n=0;
for i=1:Nbit
for k=1:((N/2)+1)
    if N==0
      output=[set0,set1];

      n=n+1
    else
        set0(1,k)=array(1,j);
        set1(1,k)=array(1,j+1);
%        if k>(n+1)
```

```
%       l=fix(k/(n+1));
%       else
%           l=1;
%       end
        set0(i+1,k)=0;
        set1(i+1,k)=1;
    end
    j=j+2;
    N=N-2;
end
N=Nlength;
j=1;
end
End
```

Myrecursive function to assign peak indexing value:

```
function f=myrecursive(array)
%  List_peaks = 1E-06*[2.5 2.62 2.8 2.9 2.98 3.4 3.52 3.6 3.72 3.79 3.85 4.0 3.46 3.15 2.75 3.33];
% load(new_pgm.m,List_peaks);
% Nlength=length(List_peaks);
N=length(array);
Nbit=log2(N);
f=zeros((Nbit+1),N);
set0=zeros((Nbit+1),(N/2));
set1=zeros((Nbit+1),(N/2));
% array=List_peaks;
```

```
bit=zeros(Nbit,N);

j=1;n=0;

for k=1:((N/2)+1)

   if N==0

     f=[set0,set1];

     array=f;

     n=n+1

   else

       set0(1,k)=array(1,j);

       set1(1,k)=array(1,j+1);

       set0(2,k)=0;

       set1(2,k)=1;

  end

   j=j+2;

   N=N-2;

End
```

Outputs for this code:

number of traps =4

1) Assigning of peak index for each peak:

2.5	**2.62**	**2.8**	**2.9**	**2.98**	**3.40**	**3.52**	**3.6**	**3.72**	**3.79**	**2.85**	**4.0**	**3.46**	**3.15**	**2.75**	**3.33**
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1

Code for calculating amplitude and time constant ratio for each trap:

```
%//code for caluculating amplitude and time constant ratio of each trap//

dummy0=0;
  dummy1=0;
  new_set0=zeros(Nbit,(Nvalue/2)); % new sets to take index value is zero and index value one seperately
  new_set1=zeros(Nbit,(Nvalue/2));
  timeconstant_ratio_trap=zeros(1,Nbit);
  amplitude=zeros(Nbit,(Nvalue/2));
  j=1;l=1;
  for i=1:1:Nbit
  for k=1:1:Nvalue
     if res((i+1),k)==0
        dummy0=dummy0+res(1,k);
        new_set0(i,j)=res(1,k);
        j=j+1;
```

```
        if j==((Nvalue/2)+1)
          j=1;
        end
      else if res((i+1),k)==1
        dummy1=dummy1+res(1,k);
        new_set1(i,l)=res(1,k);
        l=l+1;
        if l==((Nvalue/2)+1)
          l=1;
        end
        end
      end

    End
 timeconstant_ratio_trap(1,i)=(dummy0/dummy1); %output ratio of two time constants for each
trap
      amplitude=new_set0-new_set1;
    end
    for i=1:Nbit
      for k=1:(Nvalue/2)
        if amplitude(i,k)<0
          amplitude(i,k)=-amplitude(i,k); % output amplitude for each trap
        end
      end
    end
```

Output for amplitude and time constant ratio for each traps from 16-level RTN:

1) RTN Amplitude for each trap:

Trap-1	0.12	0.1	0.42	0.08	0.07	0.15	0.31	0.58
Trap-2	0.3	0.28	0.54	0.2	0.13	0.21	0.71	0.18
Trap-3	0.48	0.78	0.72	0.7	0.26	0.64	1.1	0.67
Trap-3	1.22	1.17	1.05	1.1	0.48	0.25	0.77	0.27

2) Time constant ratio for each trap:

Trap	Time constant ratio
Trap-1	0.9548
Trap-2	0.9563
Trap-3	0.9705
Trap-3	0.9436

CONCLUSION

An apparatus and method for characteristic physical constant extraction for determining multiple trap defects in a semiconductor device, including receiving signal representing a change in a drain current of the semiconductor device over time, the signal comprising a time series of values, constructing a histogram representation, each peak being associated with a state of a Random Telegraph Noise (RTN) signal caused by a plurality of bistable traps of mobile charge carriers in the semiconductor device. Assigning a peak index value for each peak; calculating an amplitude and a time constant ratio for each trap; determining theoretical intensities for each peak; and determining a number of traps based on a correspondence between the calculated amplitude and time constant ratio for each trap and the theoretical intensity for each peak associated with each trap. Embodiments of an apparatus for extracting characteristic physical constants from a complex Random Telegraph Noise (RTN) signal includes a first stage configured to resolve a number of stable states in a time series, the stable states being associated with a plurality of defects, the time series comprising a plurality of data points, and the first stage also configured to determine an activation status of each defect; and a second stage configured to calculate a transition preference table based on physically allowed transitions, and to uniquely assign each data point of the time series to one of the stable states using the transition preference table, each defect comprising a bistable trap of a mobile charge carrier in the semiconductor device, and the plurality of defects causing Random Telegraph Noise (RTN) in a voltage threshold of one or more gates of the semiconductor device.

REFERENCES

[1] M. Kirton and M. Uren, “Noise in solid-state microstructures: A new perspective on individual defects, interface states and low-frequency (1/ƒ) noise,” Advances in Physics, vol. 38, no. 4, pp. 367–468, 1989, doi: 10.1080/00018738900101122.

[2] K. S. Ralls, W. J. Skocpol, L. D. Jackel, R. E. Howard, L. A. Fetter, R. W. Epworth, and D. M. Tennant, “Discrete resistance switching in submicrometer silicon inversion layers: Individual interface traps and low-frequency (1 f ?) noise,” Phys. Rev. Lett., vol. 52, pp. 228–231, Jan 1984, doi:10.1103/PhysRevLett.52.228.

[3] M. Luo, R. Wang, S. Guo, J. Wang, J. Zou, and R. Huang, “Impacts of random telegraph noise (RTN) on digital circuits,” IEEE Transactions on Electron Devices, vol. 62, no. 6, pp. 1725–1732, June 2015, doi:10. 1109/TED.2014.2368191.

[4] S. O. Toh, T. K. Liu, and B. Nikolic, “Impact of random telegraph signaling noise on SRAM stability,” in 2011 Symposium on VLSI Technology - Digest of Technical Papers, June 2011, pp. 204–205.

[5] Naoki Tega *et al*., "Impact of threshold voltage fluctuation due to random telegraph noise on scaled-down SRAM," *2008 IEEE International Reliability Physics Symposium*, Phoenix, AZ, 2008, pp. 541-546.

[6] H. Kurata, K. Otsuga, A. Kotabe, S. Kajiyama, T. Osabe, Y. Sasago, S. Narumi, K. Tokami, S. Kamohara, and O. Tsuchiya, “The impact of random telegraph signals on the scaling of multilevel flash memories,” in 2006 Symposium on VLSI Circuits, 2006. Digest of Technical Papers., June 2006, pp. 112–113, doi:10.1109/VLSIC.2006.1705335

[7] P Dutta, P M Horn, "Low-frequency fluctuations in solids: 1/f noise ,“ in *Reviews of Modern Physics*, vol. 53, no. 3, pp. 497-516, July 1981.

[8] J. P. Campbell *et al*., "Random telegraph noise in highly scaled nMOSFETs," *2009 IEEE International Reliability Physics Symposium*, Montreal, QC, 2009, pp. 382-388.

[9]K. Ito, T. Matsumoto, S. Nishizawa, H. Sunagawa, K. Kobayashi and H. Onodera, "The impact of RTN on performance fluctuation in CMOS logic circuits," 2011 International Reliability Physics Symposium, 2011, pp. CR.5.1-CR.5.4, doi: 10.1109/IRPS.2011.5784563.

[10] N. Tega et al., "Increasing threshold voltage variation due to random telegraph noise in FETs as gate lengths scale to 20 nm," 2009 Symposium on VLSI Technology, 2009, pp. 50-51.

[11] Seng Oon Toh, Y. Tsukamoto, Z. Guo, L. Jones, T. K. Liu and B. Nikolić, "Impact of random telegraph signals on Vmin in 45nm SRAM," 2009 IEEE International Electron Devices Meeting (IEDM), 2009, pp. 1-4, doi: 10.1109/IEDM.2009.5424228.

[12] S. H. Gu et al., "Read Current Instability Arising from Random Telegraph Noise in Localized Storage, Multi-Level SONOS Flash Memory," 2006 International Electron Devices Meeting, 2006, pp. 1-4, doi: 10.1109/IEDM.2006.346820.

[13] H. Miki et al., "Statistical measurement of random telegraph noise and its impact in scaled-down high-κ/metal-gate MOSFETs," 2012 International Electron Devices Meeting, 2012, pp. 19.1.1-19.1.4, doi: 10.1109/IEDM.2012.6479071.

[14]L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, Feb 1989.

[15] D. J. Frank, "Random telegraph noise : Measurement, analysis and consequences," in International Reliability Physics Symposium, 2012, Tutorial.

[16] Ghahramani, Z., Jordan, M.I. Factorial Hidden Markov Models. *Machine Learning* **29,** 245–273 (1997). https://doi.org/10.1023/A:1007425814087.

[17] F. M. Puglisi and P. Pavan, "RTN analysis with FHMM as a tool for multi-trap characterization in HfOX RRAM," 2013 IEEE International Conference of Electron Devices and Solid-state Circuits, 2013, pp. 1-2, doi: 10.1109/EDSSC.2013.6628059.

[18] H. Awano, H. Tsutsui, H. Ochi and T. Sato, "Multi-trap RTN parameter extraction based on Bayesian inference," International Symposium on Quality Electronic Design (ISQED), 2013, pp. 597-602, doi: 10.1109/ISQED.2013.6523672.

[19] T. Kitagaki, "Method for analyzing random telegraph signal and threshold level determination method therefor," Dec. 24 2013, US Patent 8615034B2.

[20] Miki, "Method of extracting a time constant from complex random telegraph signals," Jan. 27, 2011, US 2011/0022339 A1.