Lab 4

Tuesday 1st March, 2022

1. **Statistical Modeling and Parametric Variations**

   (d) I calculated the probability density function for both I_{sub} and Vth using the following MATLAB script:

```
1   trials = 5000;
2   W = 1e-7;
3   V_gs = 0.2;
4   S = 0.085;
5   V_th = zeros(1, trials);
6   I_sub = zeros(1, trials);
7
8   for i = 1:trials
9       L = normrnd(5e-8, 1e-9);
10      N_a = normrnd(2.4e18, 0.6e18);
11      V_th(i) = ((0.18+(13e-11)*sqrt(N_a))*(1-exp((-2e6)*L)))/(0.0952);
12      I_sub(i) = 100*(W/L)*10^((V_gs - V_th(i))/S);
13  end
14
15  i = histogram(I_sub,'Normalization','pdf');
16  xlabel('I_s_u_b (nA)')
17  ylabel('Probability (%)')
18  title('PDF of I_s_u_b')
19  %v = histogram(V_th,'Normalization','pdf');
20  %xlabel('V_t_h (V)')
21  %ylabel('Count')
22  %title('Histogram of V_t_h')
```

   (Note that only one histogram can be generated at a time, which is why one of them is commented; uncomment lines 18-21 and comment lines 14-17 to display V_{th} instead of I_{sub}) The generated PDF histograms are shown in figures 1 and 2.

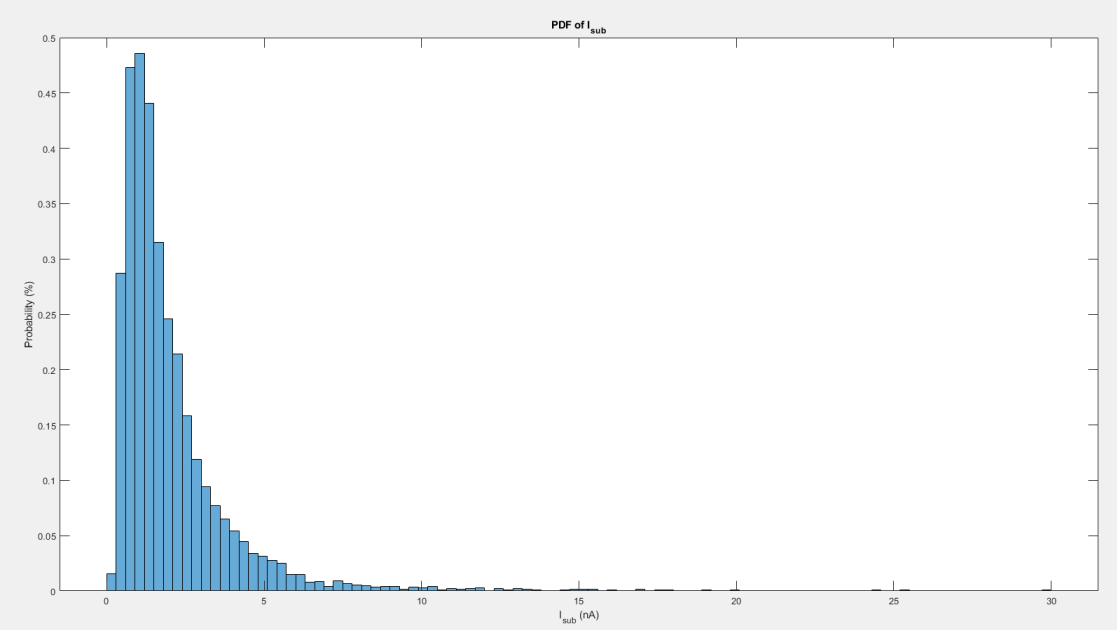

Figure 1: The generated PDF histogram for I_{sub}.

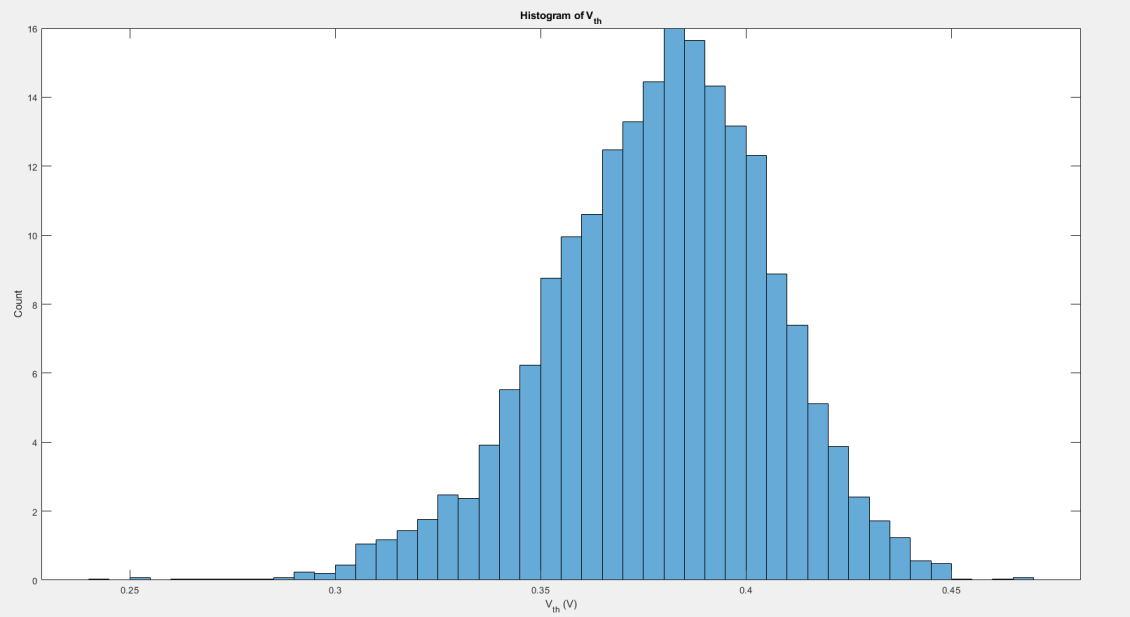

Figure 2: The generated PDF histogram for V_{th}.

(e) I altered the MATLAB code given previously to find the mean and standard deviation of the generated V_{th} values to create a new V_{th} as a Gaussian random variable. Then, I used this random variable again to generate another PDF histogram of I_{sub}, shown in figure 3.

```
trials = 5000;
W = 1e-7;
V_gs = 0.2;
S = 0.085;
V_th = zeros(1, trials);
for i = 1:trials
    L = normrnd(5e-8, 1e-9);
    N_a = normrnd(2.4e18, 0.6e18);
    V_th(i) = ((0.18+(13e-11)*sqrt(N_a))*(1-exp((-2e6)*L)))/(0.0952);
end

mu = mean(V_th);
sigma = std(V_th);
samples = 300;
I_sub = zeros(1, samples);
for i = 1:samples
    V_th_new = normrnd(mu, sigma);
    I_sub(i) = 100*(W/L)*10^((V_gs - V_th(i))/S);
end

i = histogram(I_sub,'Normalization','pdf');
xlabel('I_s_u_b (nA)')
ylabel('Probability (%)')
title('PDF of I_s_u_b using V_t_h as a random variable')
```

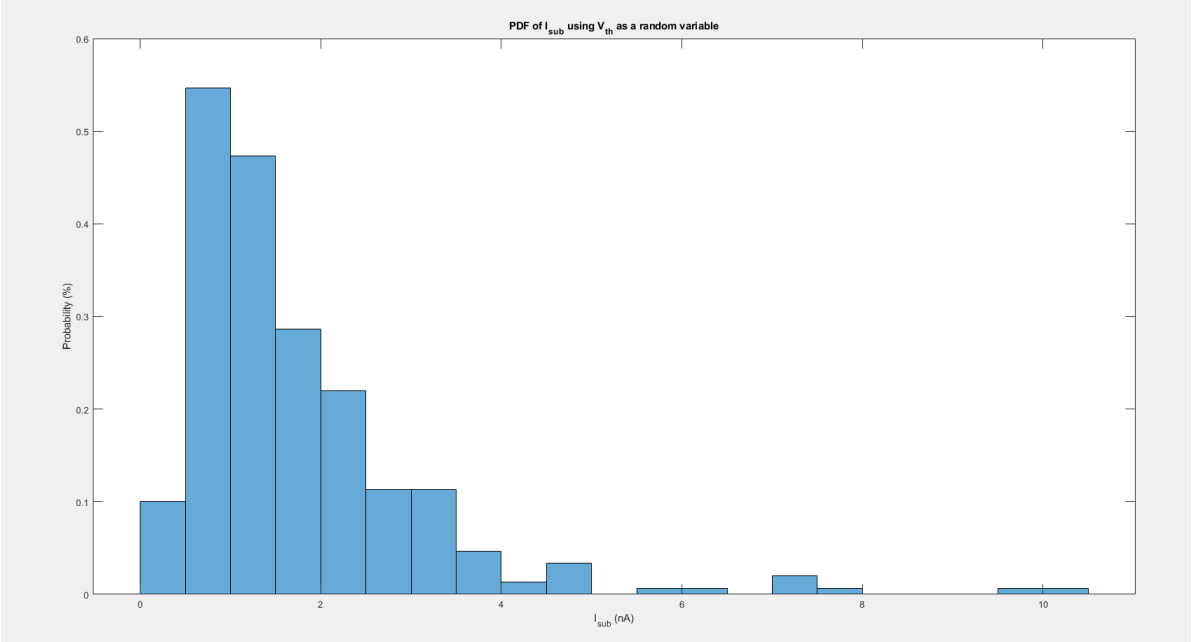

Figure 3: The new I_{sub} PDF using V_{th} as a Gaussian random variable.

I did use significantly less samples to demonstrate that a PDF of some similarity can be generated using this method. The PDF in figure 3 exhibits similar properties as the PDF in figure 1, with the major difference being the bin width (due to the fewer number of samples). Both figures seem to have a max bin height around 2 nA.

(f) PCA can be performed with MATLAB's pca(X) command. By inputting an array containing sample values of V_{th} and L as shown below, several arrays are outputted.

```
trials = 5000;
W = 1e-7;
V_gs = 0.2;
S = 0.085;
V_th = zeros(1, trials);
I_sub = zeros(1, trials);
L_samples = zeros(1, trials);

for i = 1:trials
    L = normrnd(5e-8, 1e-9);
    L_samples(i) = L;
    N_a = normrnd(2.4e18, 0.6e18);
    V_th(i) = ((0.18+(13e-11)*sqrt(N_a))*(1-exp((-2e6)*L)))/(0.0952);
    I_sub(i) = 100*(W/L)*10^((V_gs - V_th(i))/S);
end

X = [L_samples; V_th];
[W, T, V] = pca(X);
```

This code generates principle component coefficient array W, score array T, and square variance V. W is a matrix of size `trials` $\times$ 1, T is a 2×1 matrix, and V is a single value (1×1 matrix). A sample value of T I received is [-13.4406, 13.4406] and a sample value of V is 361.2983.

However, with this information, I do not know how to re-construct a Gaussian sample.

(g) The PDFs received by changing the standard distribution of the gate length to 2nm and 5 nm are shown in the figures below.

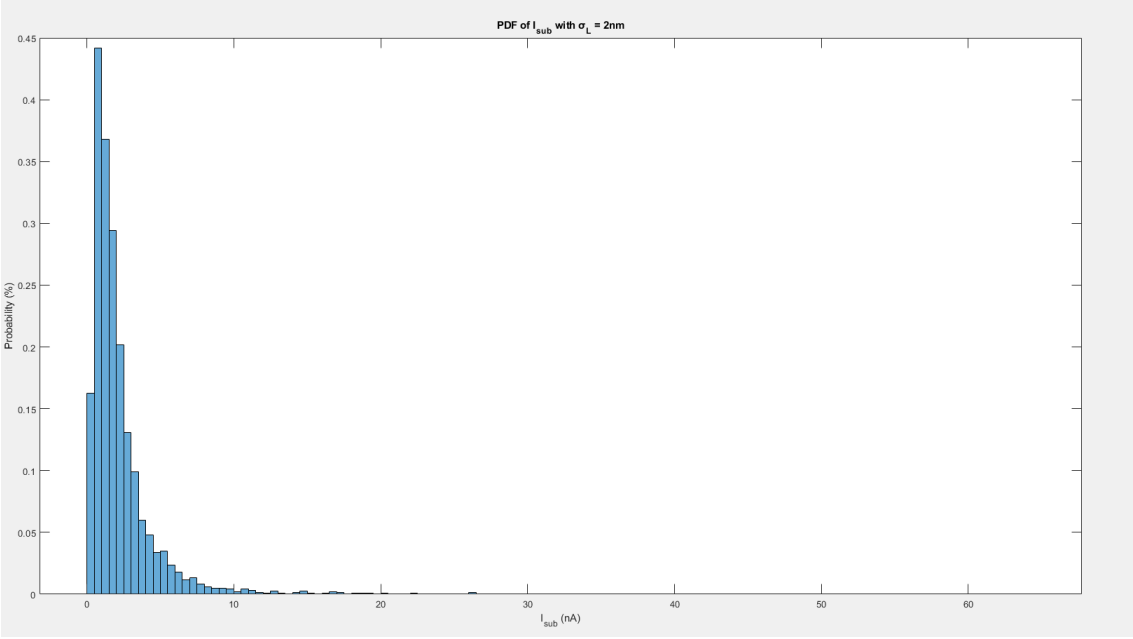

Figure 4: The generated PDF histogram for I_{sub} at $\sigma_L = 2$nm.

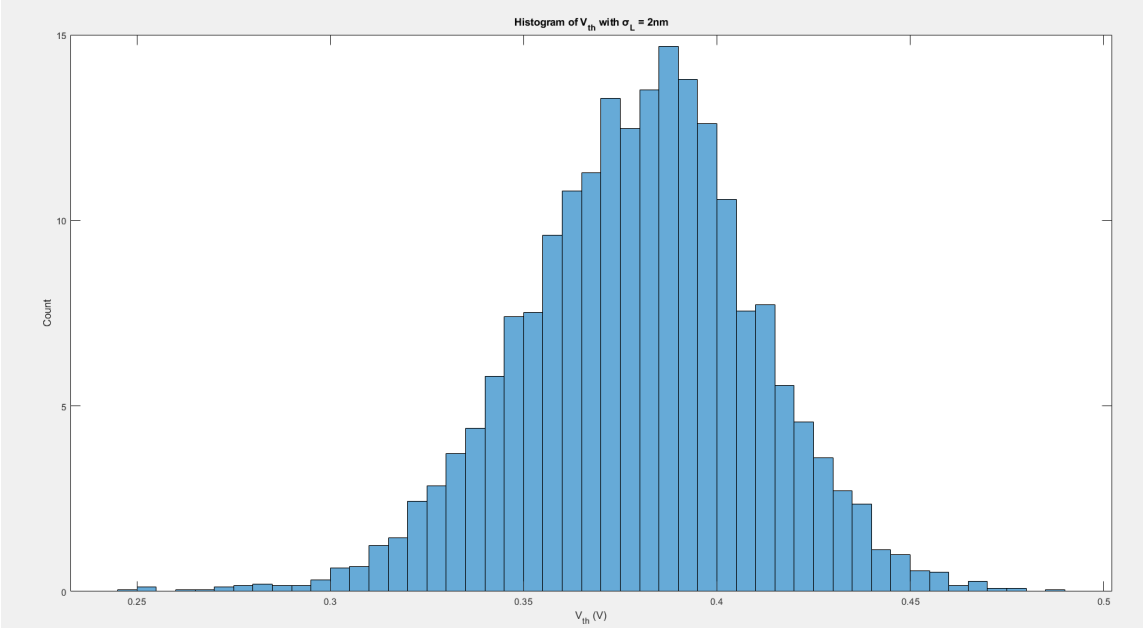

Figure 5: The generated PDF histogram for V_{th} at $\sigma_L = 2$nm.

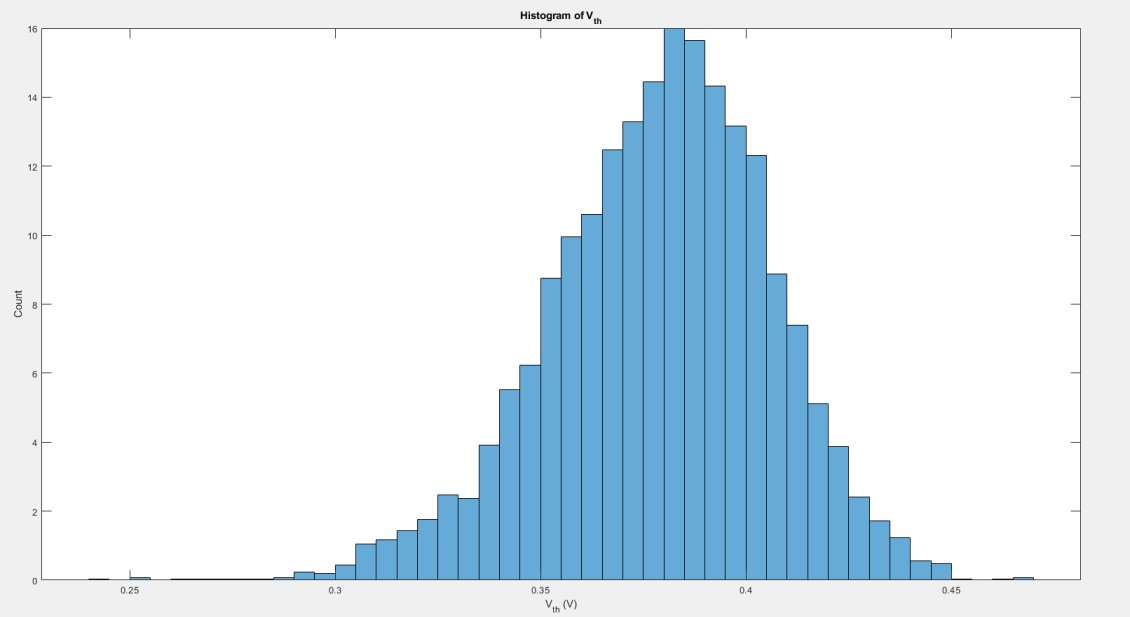

Figure 6: The generated PDF histogram for I_{sub} at $\sigma_L = 5$nm.

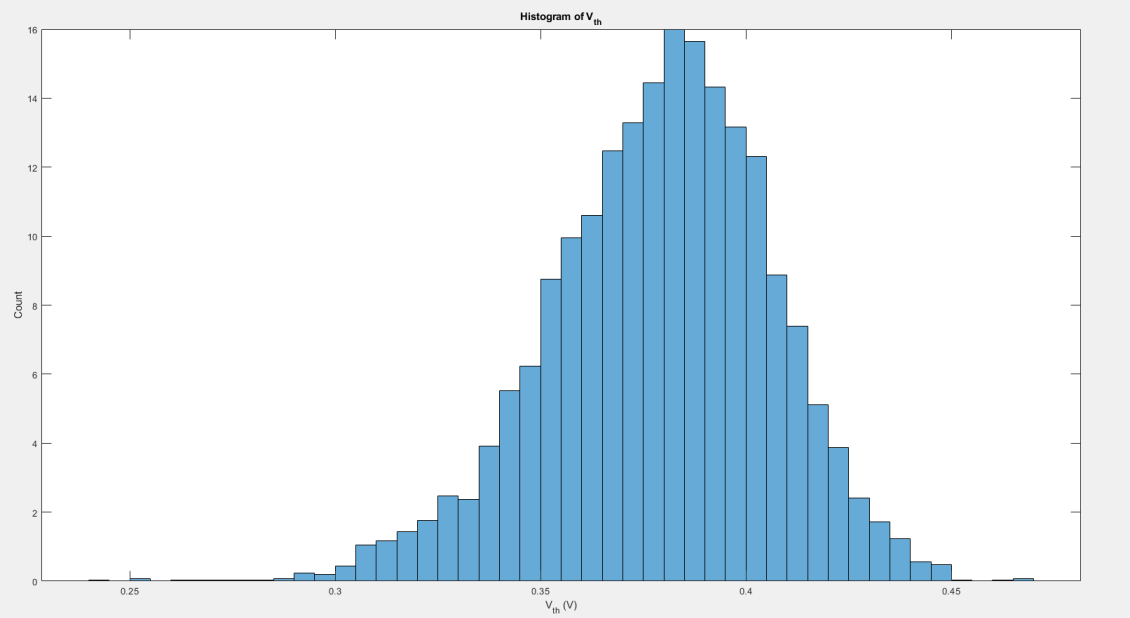

Figure 7: The generated PDF histogram for V_{th} at $\sigma_L = 5$nm.

The golden PDF already starts to differ from its original state at $\sigma_L = 5$nm, however, because I do not have a PCA PDF to compare it to, I cannot answer the second part of this prompt.

2. **Critical Area Calculator** Critical area for shorts, geometrically, is calculated by expanding the size of all the metals and finding where they overlap. In the SVRF script, this looks like:

```
LAYOUT SYSTEM GDSII
LAYOUT PATH "s1494.gds"
LAYOUT PRIMARY "s1494_bench"

PRECISION 2000
RESOLUTION 1
LAYOUT ERROR ON INPUT NO
FLAG SKEW YES
FLAG ACUTE   YES
FLAG OFFGRID YES
DRC MAXIMUM RESULTS ALL
DRC RESULTS DATABASE "CA.gds" GDSII
DRC SUMMARY REPORT "CA_CALC.rep"
DRC MAXIMUM VERTEX 199
DRC KEEP EMPTY YES
LAYER metalof 15
LAYER metalos 16
LAYER metalte 18
LAYER metaltt 22
LAYER metaltth 23
LAYER metalthf 43

METAL1 = metalof OR metalos
METEMP = metalte OR metaltt
METAL2 = METEMP OR metaltth

temp1 = SIZE METAL1 BY 0.003 OVERLAP ONLY
temp2 = SIZE METAL2 BY 0.003 OVERLAP ONLY
temp3 = SIZE metalthf BY 0.003 OVERLAP ONLY
temp4 = temp1 OR temp2
caa1 =  temp4 OR temp3

caa1 {copy caa1} DRC CHECK MAP caa1 109
DRC PRINT AREA caa1
```

The code can be modified by changing the SIZE commands to the size of the defect divided by two. Screenshots of the critical area are found in the figures below.

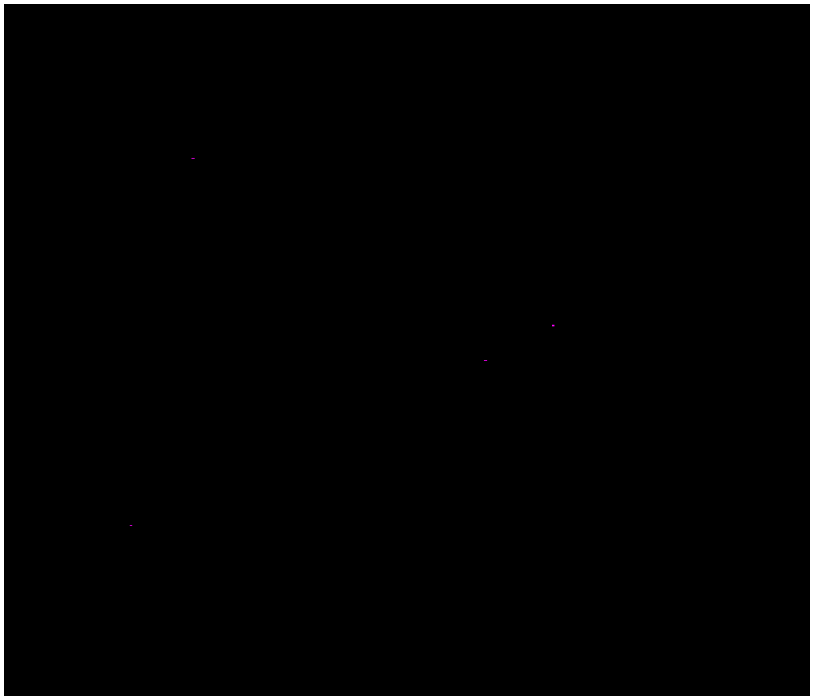

Figure 8: Critical area at a defect size of 6 µm.

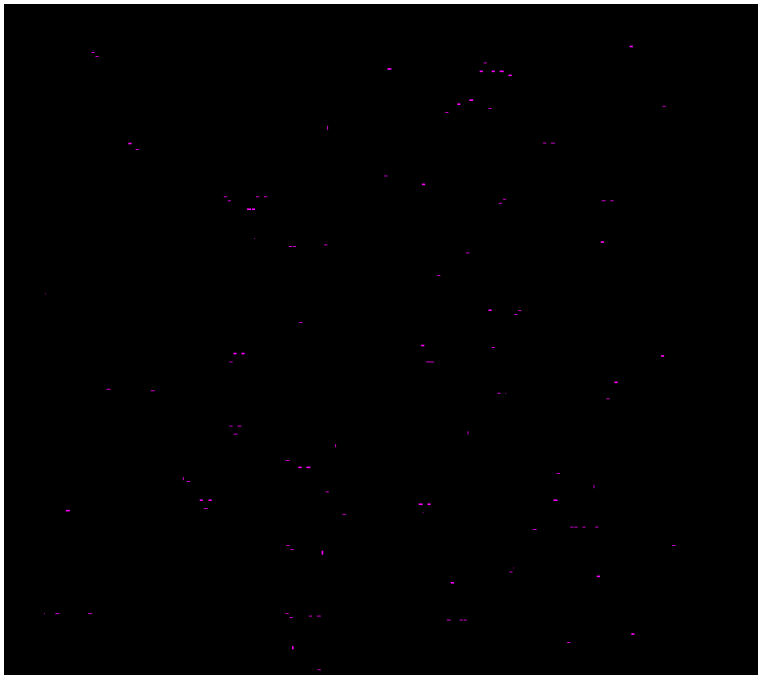

Figure 9: Critical area at a defect size of 12 µm.

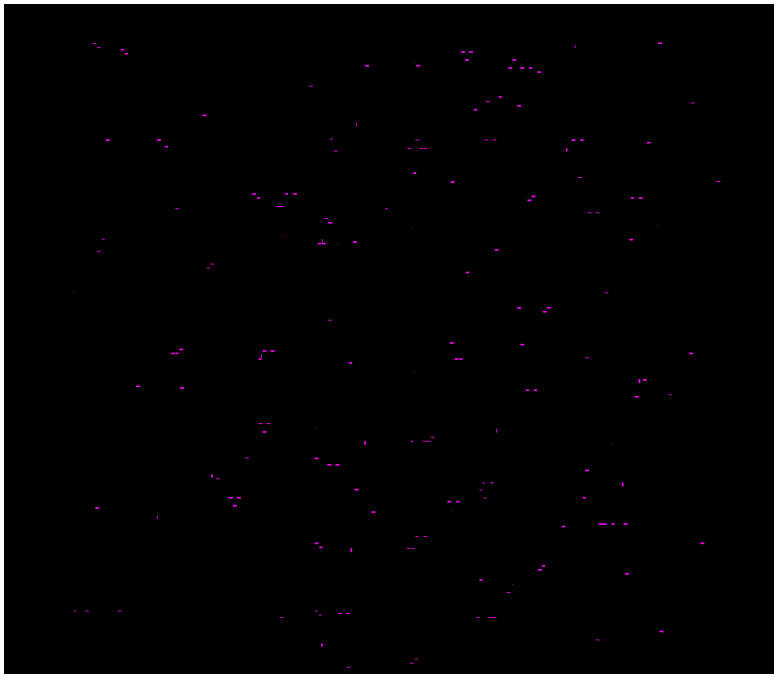

Figure 10: Critical area at a defect size of 18 µm.

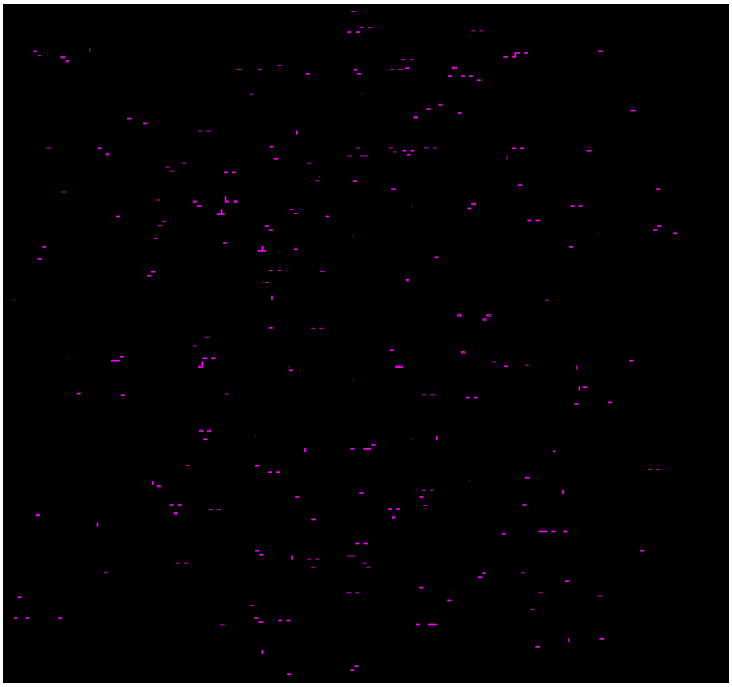

Figure 11: Critical area at a defect size of 24 µm.

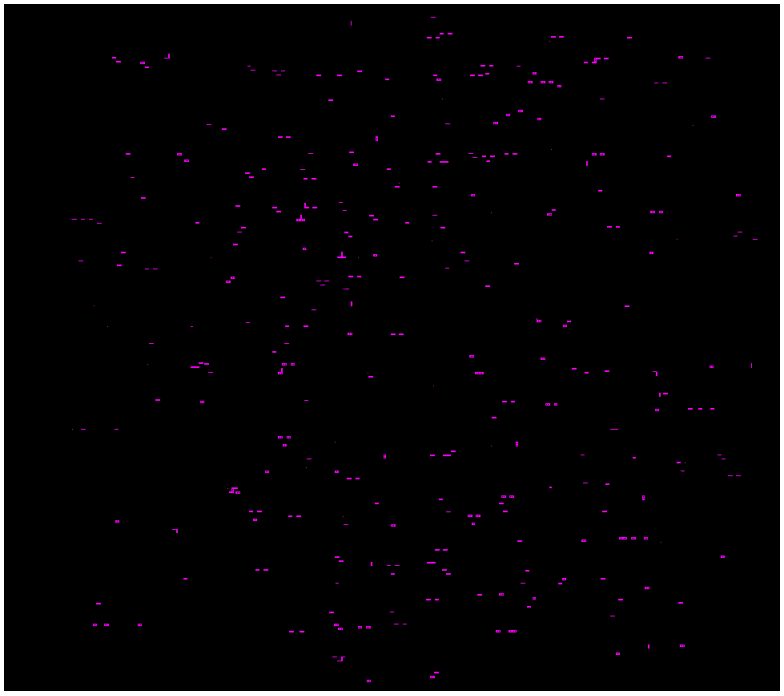

Figure 12: Critical area at a defect size of 30 µm.

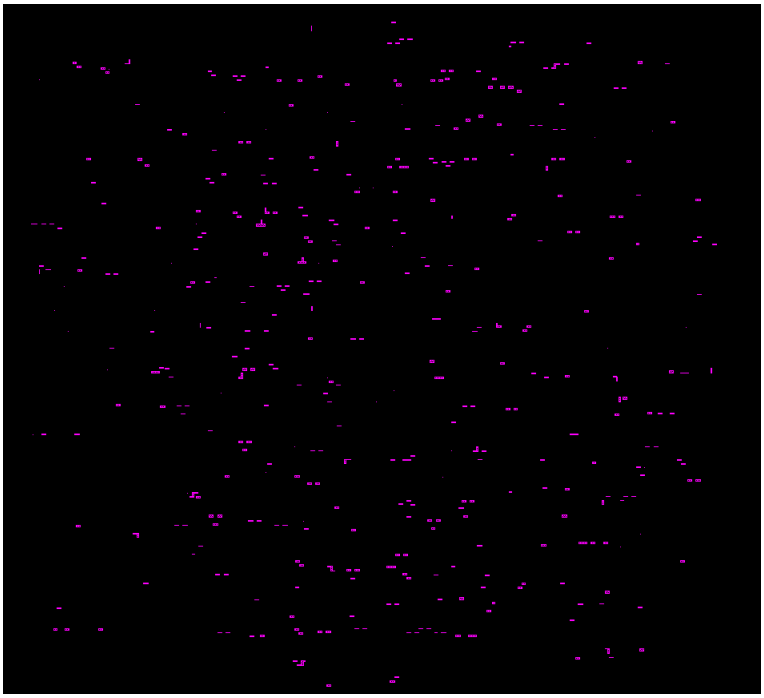

Figure 13: Critical area at a defect size of 36 µm.

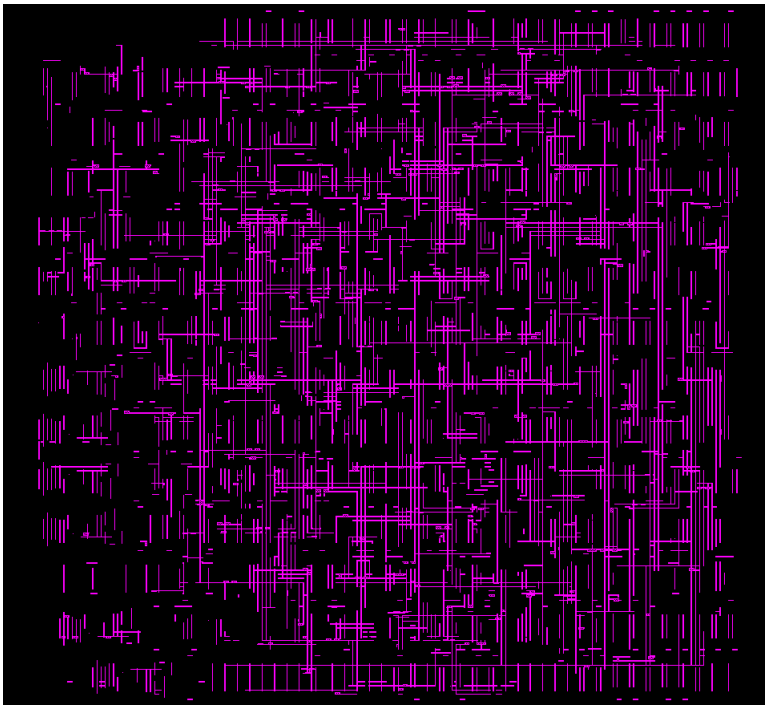

Figure 14: Critical area at a defect size of 42 µm.

Because of the `DRC PRINT AREA caa1` line, the area is printed to the console after the script is run. (This area can also be verified by running the `CALCULATE AREA` macro after selecting all objects on the layer on Calibre WORKbench.) The areas are as follows:

Defect Size (nm)	Critical Area (square microns)
6	0.000136
12	0.014234
18	0.054291
24	0.127998
30	0.247191
36	0.416053
42	5.638157

3. **Metal Fill**

To add metal (and via) fills, I used the following commands in Innovus:

```
setMetalFill -windowSize 10 10 -windowStep 10 10 -minDensity 20 -maxDensity 80
addMetalFill
setViaFill -windowSize 10 10 -windowStep 10 10 -minDensity 20 -maxDensity 80
addViaFill
verifyMetalDensity
```

After inserting metal and via fills, I found the following number of violations:

Metal Fill Minimum and Maximum Densities (Min %, Max %)	Number of Violations
20%, 80%	0
30%, 70%	7
40%, 60%	17
45%, 55%	38

It seems that as the density of fills becomes forced to be around 50%, the number of violations goes up. I can imagine that a minimum density of 49% and a maximum density of 51% would lead to the most number of violations.

The beginning layout can be found in figure 15, and some sample generated layouts containing metal and via fills can be found in figures 16 and 17.

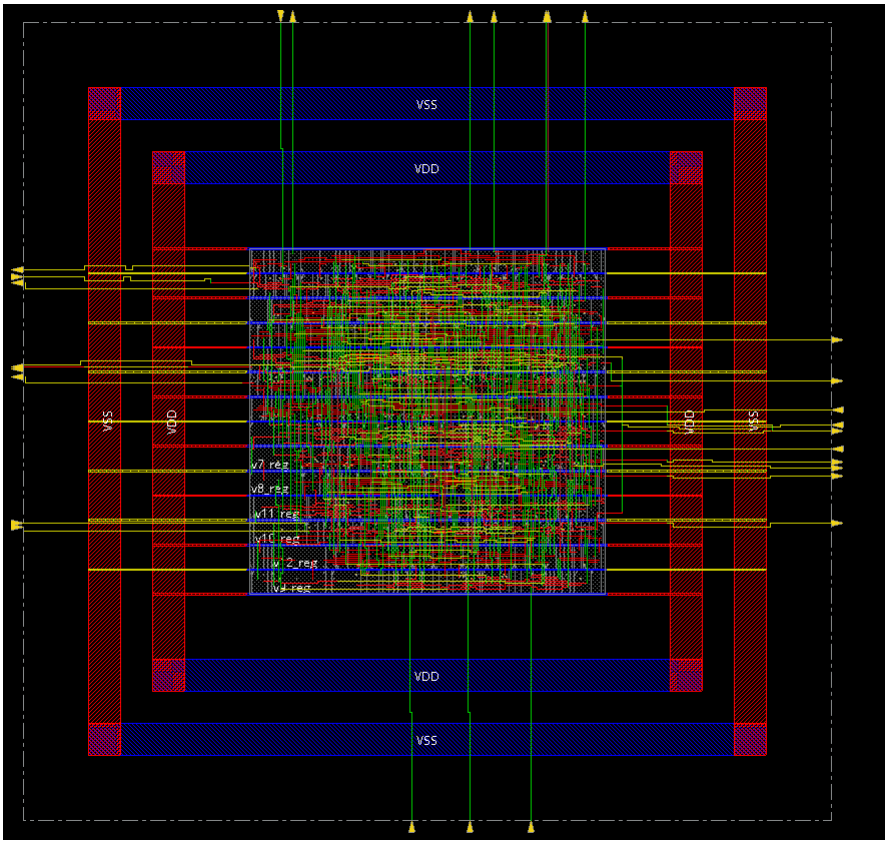

Figure 15: The clean layout.

Figure 16: The generated layout for metal/via densities at a min density of 20% and a max density of 80%.

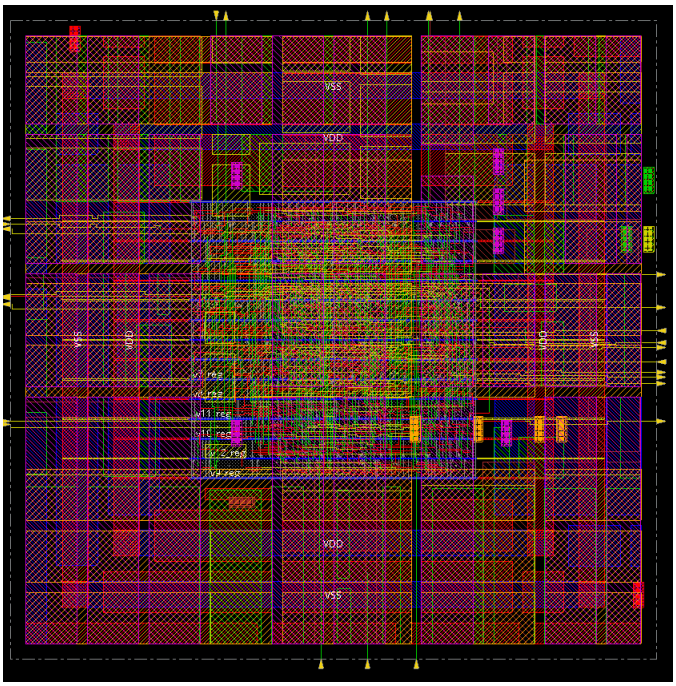

Figure 17: The generated layout for metal/via densities at a min density of 45% and a max density of 55%.