Extraction of K_n' , V_{T0} , λ parameters of Nmos transistor using 180nm Technology

AIM: Extraction of K_n' , V_{T0} , λ parameters of Nmos transistor using 180nm Technology

SOFTWARE USED: Cadence Virtuoso, Matlab

Extraction of , rameters of Nmos transistor using 180nm Technology

PROCEDURE:

- **Import the NMOS experimental data** from a CSV file containing VGS and IDS values at three different VDS levels.
- **Extract the gate voltage (VGS)** and drain currents (IDS1, IDS2, IDS3) from respective columns.
- **Extract the VDS values** used during the experiment from the file.
- **Compute the square root of IDS** for each VDS level to linearize the saturation current equation.
- **Fit a 5th-order polynomial** to the √IDS vs VGS data for each VDS using polyfit.
- **Differentiate the polynomial** using polyder to get the slope of the fitted curve.
- **Find the point of maximum slope** for each curve, which represents the most linear region in the saturation zone.
- **Construct a tangent line** at the point of maximum slope for each curve.
- **Calculate the threshold voltage (Vth)** by finding the x-intercept of each tangent line.
- **Compute the transconductance parameter (k′)** using the square of the slope at the tangent point.
- **Average the Vth and k′ values** from all three curves to get final results.
- **Plot the input characteristics** (IDS vs VGS) and **fitted lines** (√IDS vs VGS) with extracted Vth and k′ values for visualization.

Concept Used :

$$I_{DS} = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(V_{gs} - V_T)^2$$

$$\sqrt{I_{DS}} = \sqrt{\frac{K_n}{2}}V_{gs} - \sqrt{\frac{K_n}{2}}V_{T0}$$

$$y = mx + c \quad \text{where} \quad m = \sqrt{\frac{K_n}{2}}, \quad c = -\sqrt{\frac{K_n}{2}}V_{T0}$$

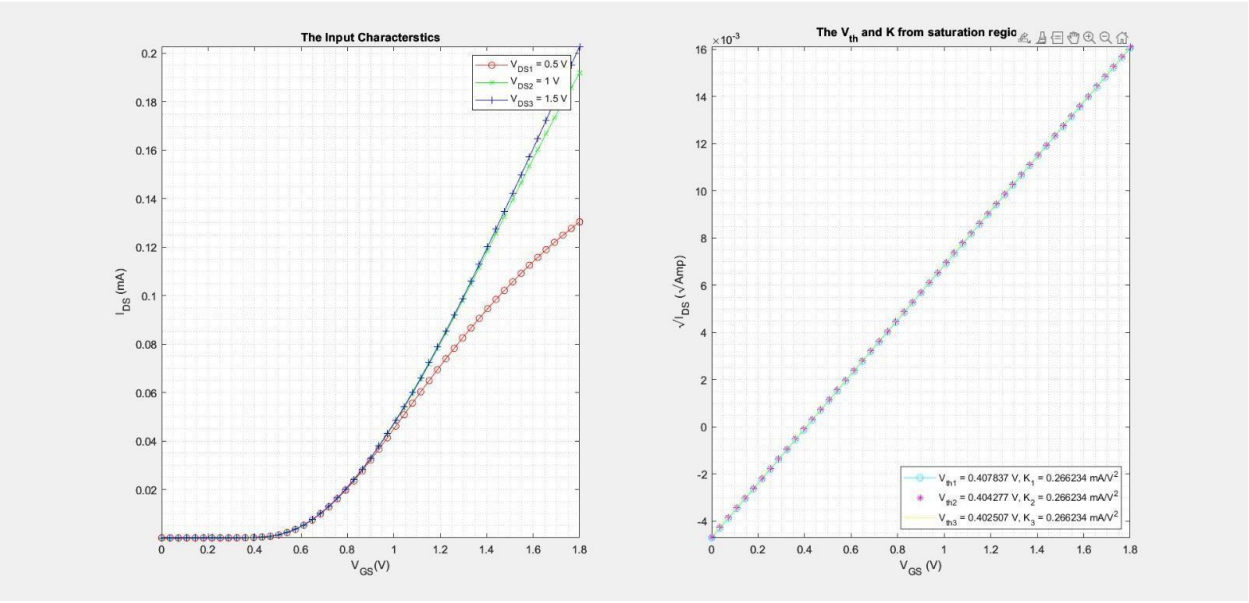
$$K_n = 2m^2 \qquad V_{T0} = -\frac{c}{m}$$

```
clc;
close all;
clear ;   % Clear command window and workspace %
name = importdata("/home/hari/nmosLongip1.csv");
Vg = abs(name(:,1));% Take gate Voltages from column 1 %

id1 = name(:,2);id2 = name(:,3);id3 = name(:,4); % Take current values from columns %

Vds1 = name(1,5);Vds2 = name(1,6);Vds3 = name(1,7); % Take drain Voltage from columns %

Order = 5; %Order of polynomial fit%

y1 = sqrt(id1);y2 = sqrt(id2);y3 = sqrt(id3); % Take square root of currents %
x = Vg;

f1 = polyfit(x,y1,Order);f2 = polyfit(x,y1,Order);f3 = polyfit(x,y1,Order); % Fit a higher order polynomial %

g1 = polyval(f1,x);g2 = polyval(f2,x);g3 = polyval(f3,x);% expressing polynomial %

u1 = polyder(f1);u2 = polyder(f2);u3 = polyder(f3); % Calculating derivative %

m1 = polyval(u1,x);m2 = polyval(u2,x);m3 = polyval(u3,x);

% Selection index based on maximum slope for Vds1 %
C1 = max(m1);
index1 = find(m1 == C1);

% Selection index based on maximum slope for Vds2 %
C2 = max(m2);
index2 = find(m2 == C2);

% Selection index based on maximum slope for Vds3 %
C3 = max(m3);
index3 = find(m3 == C3);
```

```
% Index location y axis and x axis points %
y11 = y1(index1);
x11 = x(index1);

y22 = y2(index2);
x22 = x(index2);

y33 = y3(index3);
x33 = x(index3);

% Creating lines with maximum slopes for each Vds values %
a11 = C1;
b11 = y11-(C1*x11);
yd1 = a11*x + b11;

a21 = C2;
b21 = y22-(C2*x22);
yd2 = a21*x + b21;

a31 = C3;
b31 = y33-(C3*x33);
yd3 = a31*x + b31;
```

```
% Finding Vth and k' values using before created lines %
vth1 = (b11)/(-a11); k1 = 2*power(a11,2)*1000;
vth2 = (b21)/(-a21); k2 = 2*power(a21,2)*1000;
vth3 = (b31)/(-a31); k3 = 2*power(a31,2)*1000;

% Taking average of of Vth and k' values %
vth = mean([vth1,vth2,vth3]);
k = mean([k1,k2,k3]);

fprintf('Vth(mean) = %g V,\tK(mean) = %gmA/V^2',vth,k); % Printing Vth and k' values %

% Plotting input characterstics graph %
figure;
subplot(1,2,1);
plot(x, id1*1000, 'ro-', x, id2*1000, 'gx-', x, id3*1000, 'b+-');

legend(sprintf('V_{DS1} = %g V',Vds1),sprintf('V_{DS2} = %g V',Vds2), sprintf('V_{DS3} = %g V',Vds3));

xlabel('V_{GS}(V)'); ylabel('I_{DS} (mA)'); title('The Input Characterstics');
axis tight;
grid minor;

% Plotting obtained lines corresponding to each VDS value graphs %
subplot(1,2,2),plot(x,yd1,'co-',x,yd2,'m*', x,yd3,'y-');
legend( ...
    sprintf('V_{th1} = %g V, K_1 = %g mA/V^2', vth1, k1), ...
    sprintf('V_{th2} = %g V, K_2 = %g mA/V^2', vth2, k2), ...
    sprintf('V_{th3} = %g V, K_3 = %g mA/V^2', vth3, k3), ...
    'Location', 'northeast' ...
);

xlabel('V_{GS} (V)');
ylabel('\surdI_{DS} (\surd{Amp})');
title('The V_{th} and K from saturation region');
axis tight;
grid minor;
```

Output Graphs :



Output :

```
Vth(mean) = 0.404874 V, K(mean) = 0.266234mA/V^2
```

Lambda (λ) Value Extraction from Short Channel NMOS :

Formula :

$$\lambda = \frac{I_2 - I_1}{V_1 I_2 - V_2 I_1}$$

I_1 is current for $V_{ds} = V_1 = V_{ov}$

I_2 is current for $V_{ds} = V_2 = V_{dd}$

Code :

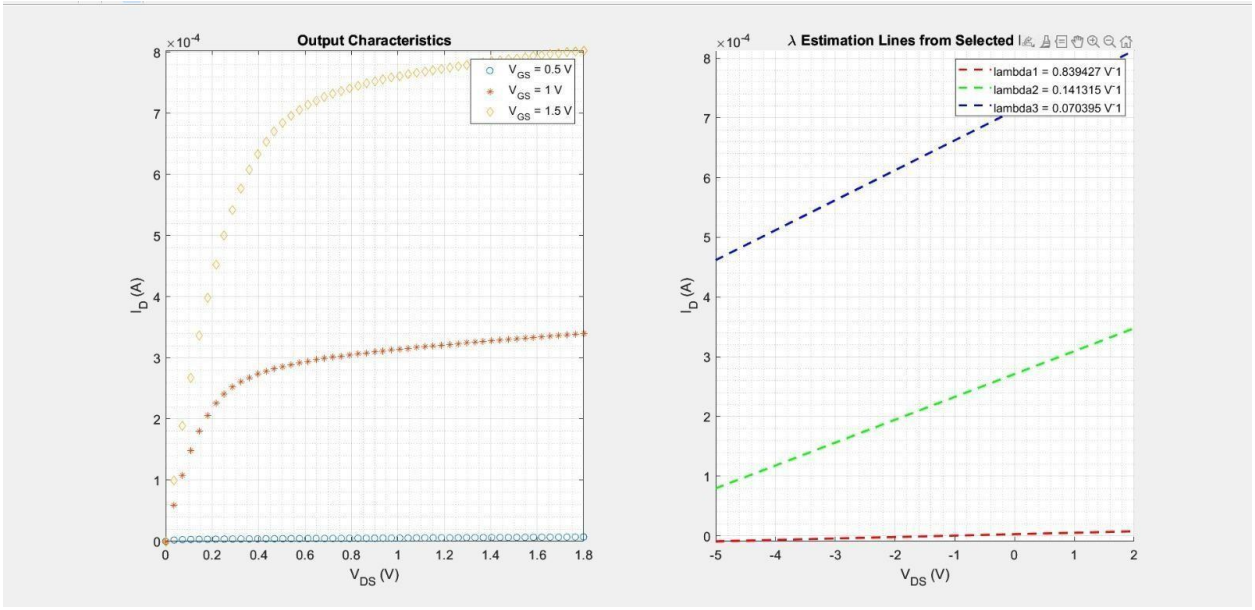
```
% Load data
name1 = importdata("/home/hari/nmos_outchar3.csv");
Vds = abs(name1(:,1));

id1 = name1(:,2); id2 = name1(:,3); id3 = name1(:,4);
Vgs1 = name1(:,5); Vgs2 = name1(:,6); Vgs3 = name1(:,7);

% Set threshold voltage
vth = 0.4;

% Tolerance for finding Vds matches
tol = 1e-3;

% Sweep 1 (id1, Vgs1)
Vgs_1 = Vgs1(1);  % Assuming Vgs is constant in the column
Vov_1 = Vgs_1 - vth;

[~, idx_vov1] = min(abs(Vds - Vov_1));  % closest index to Vov
[~, idx_1p8] = min(abs(Vds - 1.8));     % closest index to 1.8V

I1 = id1(idx_vov1);
I2 = id1(idx_1p8);
V1 = Vds(idx_vov1);
V2 = Vds(idx_1p8);
lambda1 = (I2 - I1) / (V2 * I1 - V1 * I2);

vds_line = linspace(-5, 2, 500);

m1 = (I2 - I1) / (V2 - V1);
c1 = I1 - m1 * V1;
line1 = m1 * vds_line + c1;
```

```
% Sweep 2 (id2, Vgs2)
Vgs_2 = Vgs2(1);
Vov_2 = Vgs_2 - vth;

[~, idx_vov2] = min(abs(Vds - Vov_2));
I1 = id2(idx_vov2);
I2 = id2(idx_1p8);
V1 = Vds(idx_vov2);
V2 = Vds(idx_1p8);
lambda2 = (I2 - I1) / (V2 * I1 - V1 * I2);
m1 = (I2 - I1) / (V2 - V1);
c1 = I1 - m1 * V1;
line2 = m1 * vds_line + c1;


% Sweep 3 (id3, Vgs3)
Vgs_3 = Vgs3(1);
Vov_3 = Vgs_3 - vth;

[~, idx_vov3] = min(abs(Vds - Vov_3));
I1 = id3(idx_vov3);
I2 = id3(idx_1p8);
V1 = Vds(idx_vov3);
V2 = Vds(idx_1p8);
lambda3 = (I2 - I1) / (V2 * I1 - V1 * I2);
m1 = (I2 - I1) / (V2 - V1);
c1 = I1 - m1 * V1;
line3 = m1 * vds_line + c1;

% Average lambda
lambda = mean([lambda1, lambda2, lambda3]);

fprintf('lambda = %.5f V^-1\n', lambda);
```

```
figure;

% ---- Left subplot: Output Characteristics ----
subplot(1,2,1);
plot(Vds, id1, 'o', Vds, id2, '*', Vds, id3, 'd');
grid on;
xlabel('V_{DS} (V)');
ylabel('I_D (A)');
title('Output Characteristics');
legend('V_{GS} = 0.5 V', 'V_{GS} = 1 V', 'V_{GS} = 1.5 V');
set(gca, 'FontSize', 12);
axis tight;
grid minor;
```

```
% ---- Right subplot: λ Estimation Lines ----
subplot(1,2,2);
hold on;
plot(vds_line, line1, 'r--', 'LineWidth', 2);
plot(vds_line, line2, 'g--', 'LineWidth', 2);
plot(vds_line, line3, 'b--', 'LineWidth', 2);
grid on;
xlabel('V_{DS} (V)');
ylabel('I_D (A)');
title('\lambda Estimation Lines from Selected Points');
set(gca, 'FontSize', 12);

legend( ...
    sprintf('lambda1 = %g V^-1',lambda1), ...
    sprintf('lambda2 = %g V^-1',lambda2), ...
    sprintf('lambda3 = %g V^-1',lambda3), ...
    'Location', 'northeast' ...
);
axis tight;
grid minor;
```

Ouput Graphs :




```
lambda = 0.35038 V^-1
```

Result :

K_n' = 266 $\frac{\mu A}{V^2}$

V_{T0} = 0.40 V λ = 0.35 V^{-1}