

Total:**5.33/8**

please have a close look at the comments and endeavor to understand where you made mistakes or could have improved.

Mandatory laboratory exercise 1: Characterizing MOSFETs

Klaudia Pawlak, Suhayka Mohamed and Isak Kvanneid

IN3170 - Microelectronics

Spring 2021

Introduction

The goal of this lab exercise is to characterize off-the-shelf MOS-FET transistors. In the exercise we will be using Cadence to simulate CMOS processes and reverse engineer the device parameters for the standard nMOS and pMOS FETs of that process.

Tools

We are using Cadence to simulate the circuit and MATLAB for plotting the results.

Simulation

Task 1

In our simulation we will be using an nFET transistor with the width and length being three times the minimum length: $W = L = 540 \text{ nm}$. We will present three I_D vs V_{GS} curves where we sweep V_{GS} from $0V$ to $1.2V$ at a fixed V_{DS} value.

First we are drawing the circuit. We are using Cadence Schematic Editor L for our simulation. By pressing the "I" on the keyboard we can add an instance to our circuit. We get this window as a result:

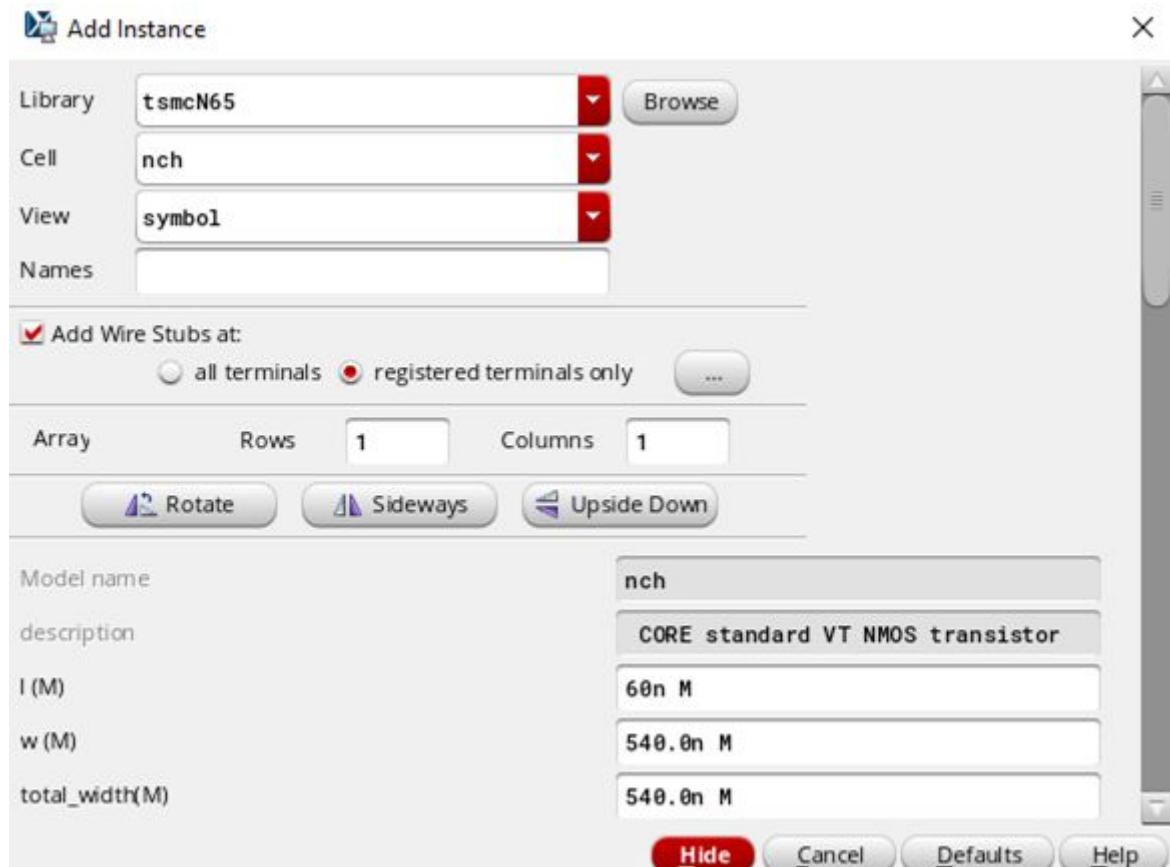
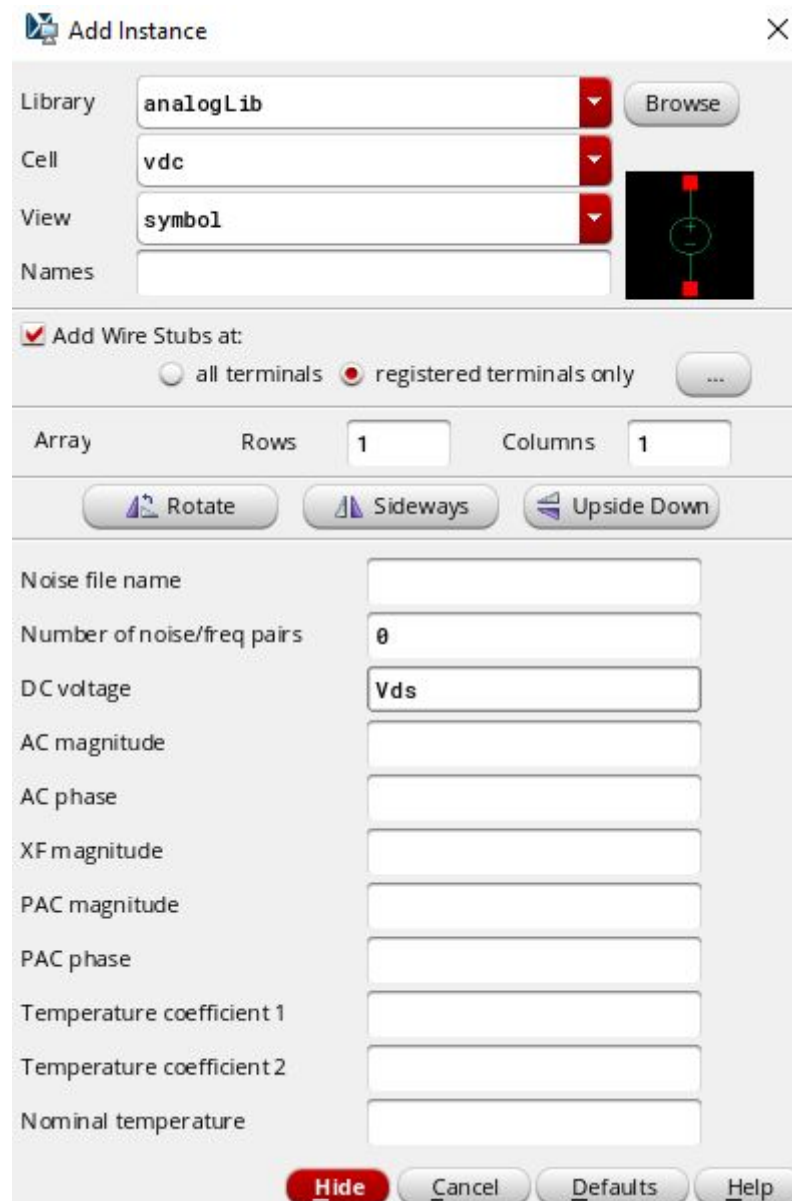


Figure 1. "Add Instance"- window for the nFET transistor

We are selecting the library "tsmcN65", cell "nch" and the view "symbol". In the w(m) field we put a value of 540 nm (our width) The rest of the parameters should change accordingly. Then we place our nFET transistor on our circuit sketch. We are using the same method to add the V_{GS} and V_{DS} to the circuit.



Add Instance

Library: **analogLib** Browse

Cell: **vdc**

View: **symbol**

Names:

☒ Add Wire Stubs at:
☐ all terminals ☒ registered terminals only ...

Array: Rows **1** Columns **1**

Rotate Sideways Upside Down

Noise file name:

Number of noise/freq pairs: **0**

DC voltage: **Vds**

AC magnitude:

AC phase:

XF magnitude:

PAC magnitude:

PAC phase:

Temperature coefficient 1:

Temperature coefficient 2:

Nominal temperature:

Hide Cancel Defaults Help

Figure 2. “Add Instance”- window for the V_{DS}

Next we are selecting the library “analogLib”, cell “vdc” and view “symbol”. We go to the cell named DC voltage and set it to “Vds”. This connects it to a variable that can be changed later on. And we are doing the same with V_{GS} , but now in the DC voltage cell we are writing “Vgs”

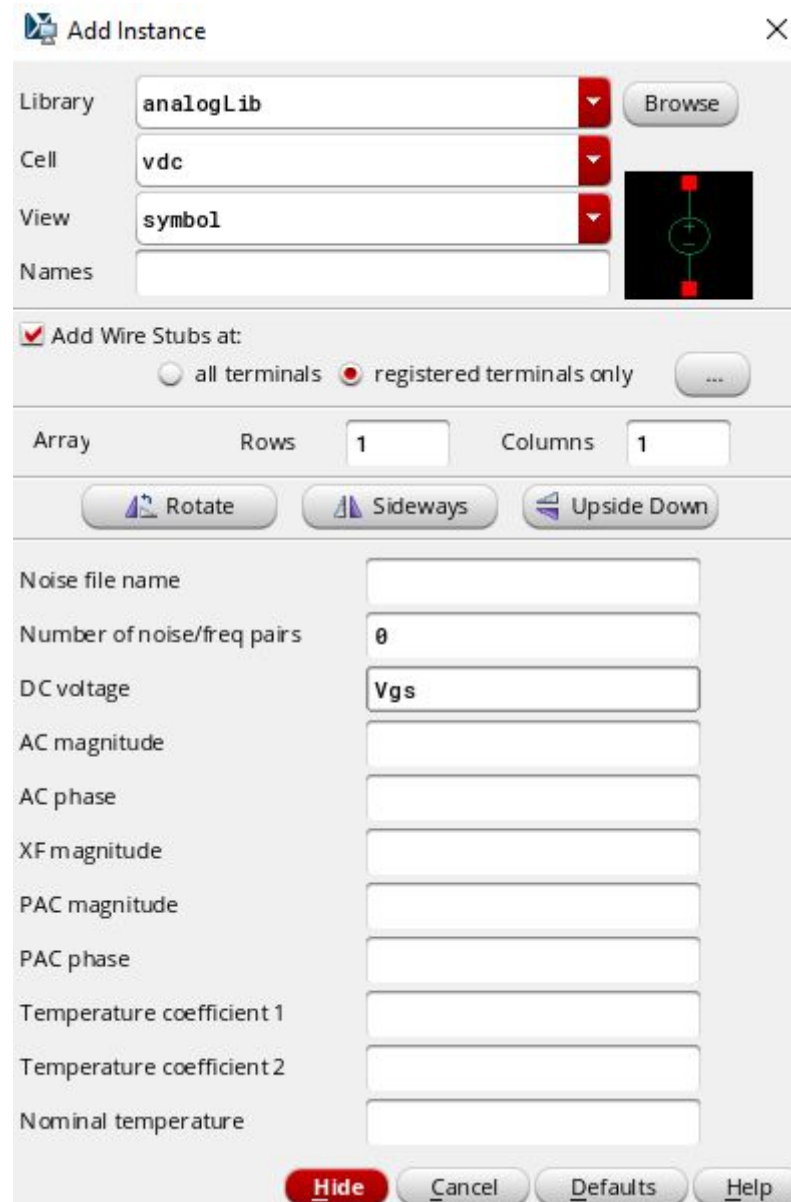


Figure 3. “Add Instance”- window for the V_{GS}

The last thing that we are missing is ground. We are using the same library and view, but in the cell field we are choosing “gnd”. See the attached picture below

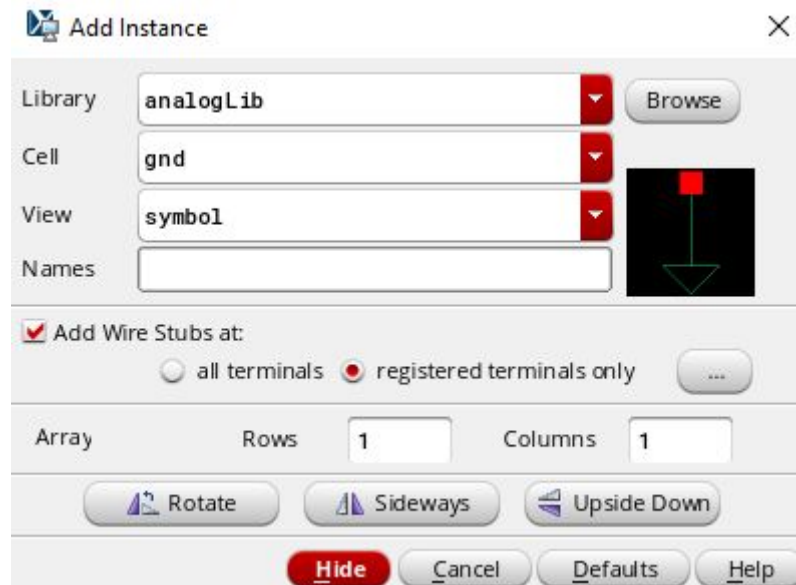
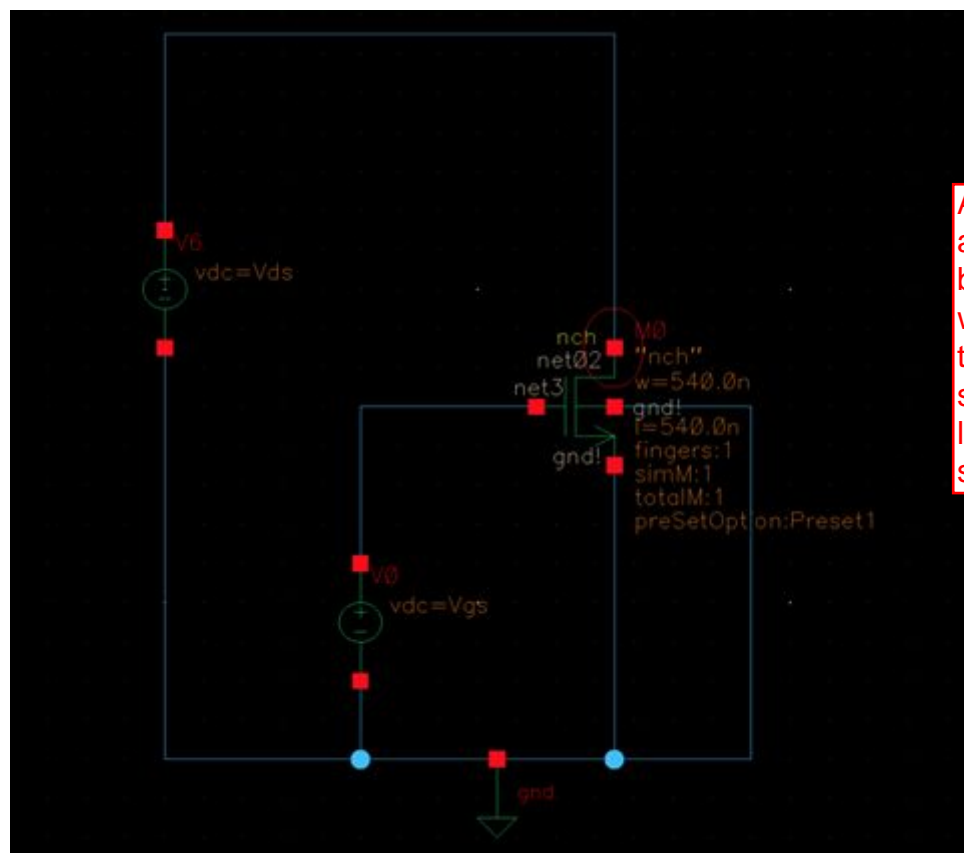


Figure 4. "Add Instance"- window for the ground

Now by pressing on the "w" on the keyboard we can add the wires to our circuit. As a result we get the following circuit.



A bit much detail to arrive at this point but nothing wrong with that, other than I tend to just skip over it: I just look at the schematics.

Figure 5. MOSFET circuit

To simulate the I_D vs V_{GS} curves we need to launch ADE L and get the ADE L window. This option can be found in the menu on the top. In the ADE L window in the column to the left under “Design Variables” we left click and go to “Copy to Cellview”. We can now choose values for our V_{GS} and V_{DS} . To plot the outputs we got to “Outputs” and then choose “To Be Plotted” and then we select the transistor on the schematics. We go to “Analysis” and “choose” a dc analysis. In this window “Choosing Analyses..” we can also sweep for the parameters we want. We can use the green button in the right menu to run the simulation. Our option should look like on the two pictures below:

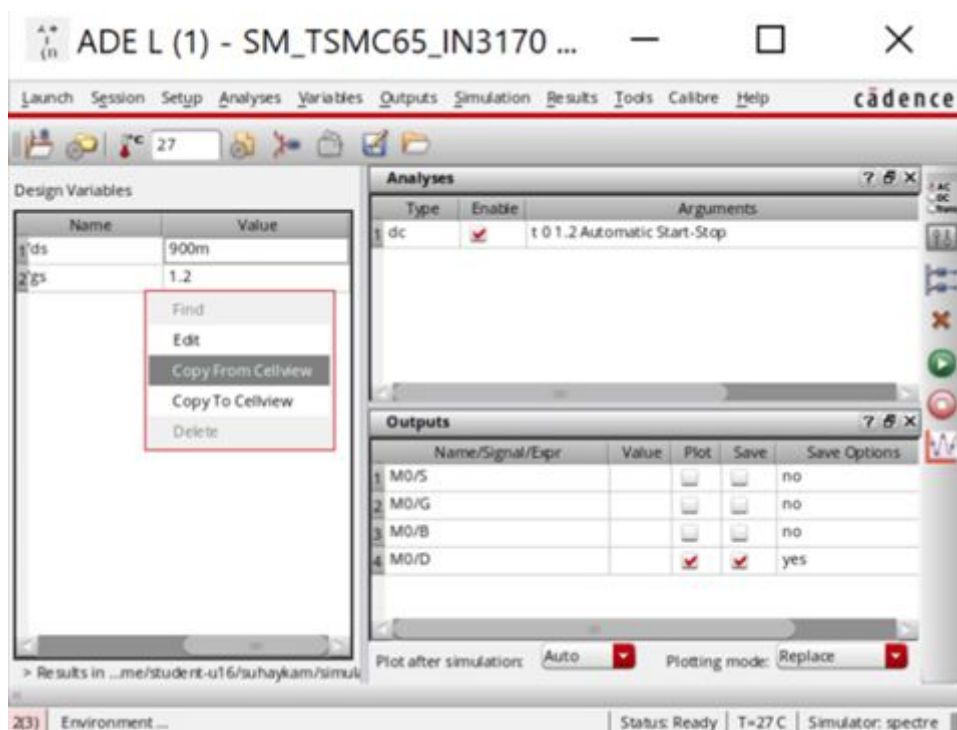


Figure 6. ADE L window



Figure 7. "Choosing Analyses" window

We will present three I_D vs V_{GS} curves where we sweep V_{GS} from $0V$ to $1.2V$. For the first curve we will choose a constant V_{DS1} such that the curve remains in the triode region. In the triode region, also known as the "linear" region is dependent on $V_{DS} < V_{SAT}$, where V_{SAT} is different for which inversion the transistor is in. For weak inversion $V_{SAT} \approx 4V_T$, V_T is the thermal voltage $V_T = 26mV$. For strong inversion the $V_{SAT} = V_{OV}$. We chose to use the condition for weak inversion where $V_{SAT} \approx 4V_T$. $V_{DS1} < V_{SAT} = 4 * 26 mV = 0.104mV$. We chose $900 mV$ for V_{DS1} .

For V_{DS2} we need a voltage such that the curve starts in the active region but enters the triode region at some point. We used the formula $V_{GS} = V_{TN} - V_{OV}$. In the simulation we estimated roughly the threshold voltage to $0.35V$. This gives us a $V_{OV} = 0.85V$. From lecture pages (page.13, chapter 5), we have the four possible

combinations for regions of operation. If we use a voltage of $0.5V$ we get the desired curve.

For V_{DS3} we need a voltage such that our curve is in the active region. We chose $V_{DS3} = 1.5V$.

We are changing the V_{DS} in the ADE L window for the values that we have found. After simulating all the curves for different V_{DS} we can export the data from Cadence to MATLAB. We right click on our curve and select the "Send to" option, and then "Export":

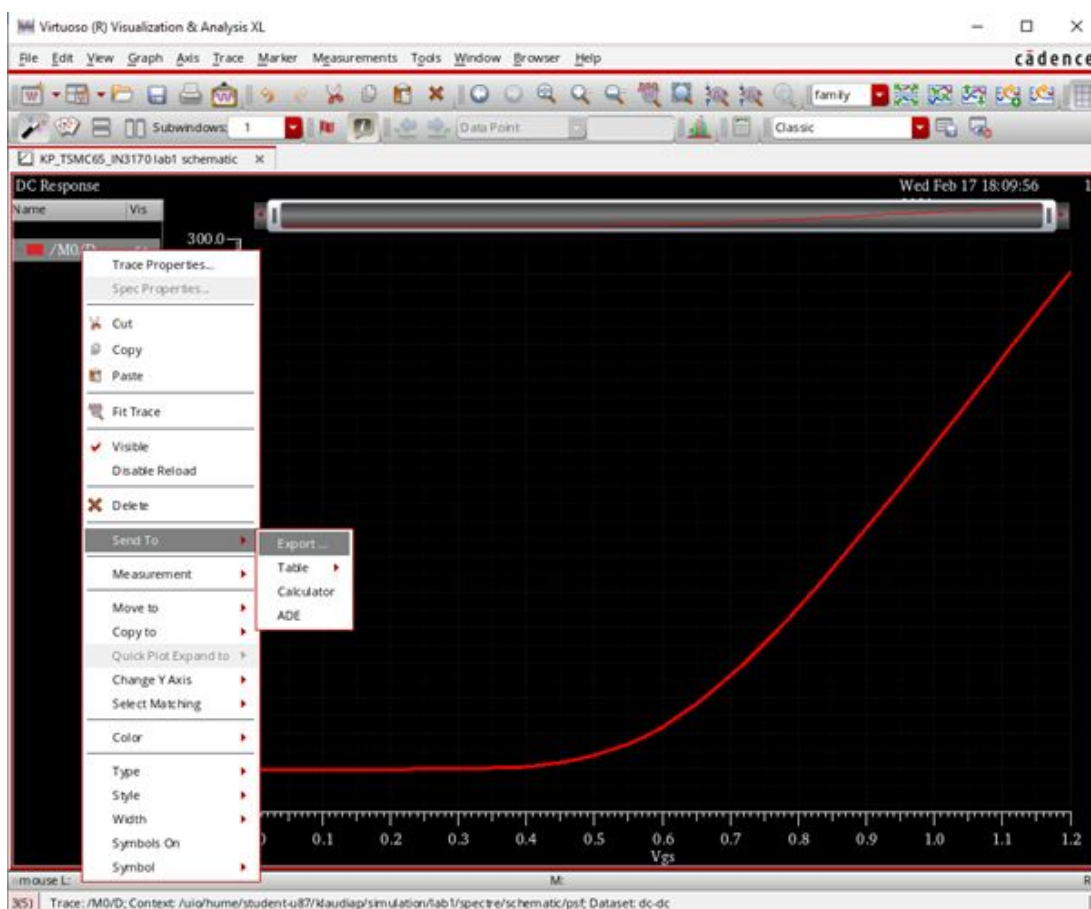


Figure 8. Exporting the data from Cadence to MATLAB

We are using those data to plot them all together. One plot with a logarithmic y-axis so we can see the weak inversion part more clearly and one with a linear y-axis so we can see the strong inversion part. The MATLAB code for reading the data and plotting the curves:


```
1. %Plotting the id vs vgs plot for different values for vds
2. data900 = csvread('task1_part1_900.csv',1); % Read the data
3. data500 = csvread('task1_part1_500.csv',1);
4. data1_5 = csvread('task1_part1_1_5.csv',1);
5.
6. figure()
7. hold on;
8. plot(data900(:,1),data900(:,2))
9. plot(data500(:,1),data500(:,2))
10. plot(data1_5(:,1),data1_5(:,2))
11. title("i_{D} vs v_{GS} plot with a linear y-axis")
12. xlabel("v_{GS}")
13. ylabel("i_{D}")
14. legend("v_{ds}=900mV", "v_{ds}=500mV", "v_{ds}=1.5V")
15. hold off;
16.
17. %Plotting the curves with a logarithmic y-axis
18. figure()
19. hold on;
20. plot(data900(:,1), data900(:,2))
21. plot(data500(:,1),data500(:,2))
22. plot(data1_5(:,1),data1_5(:,2))
23. set(gca, 'YScale', 'log');
24. title("i_{D} vs v_{GS} plot with a logarithmic y-axis")
25. xlabel("v_{GS}")
26. ylabel("i_{D}")
27. legend("v_{ds}=900mV", "v_{ds}=500mV", "v_{ds}=1.5V")
28. hold off;
```

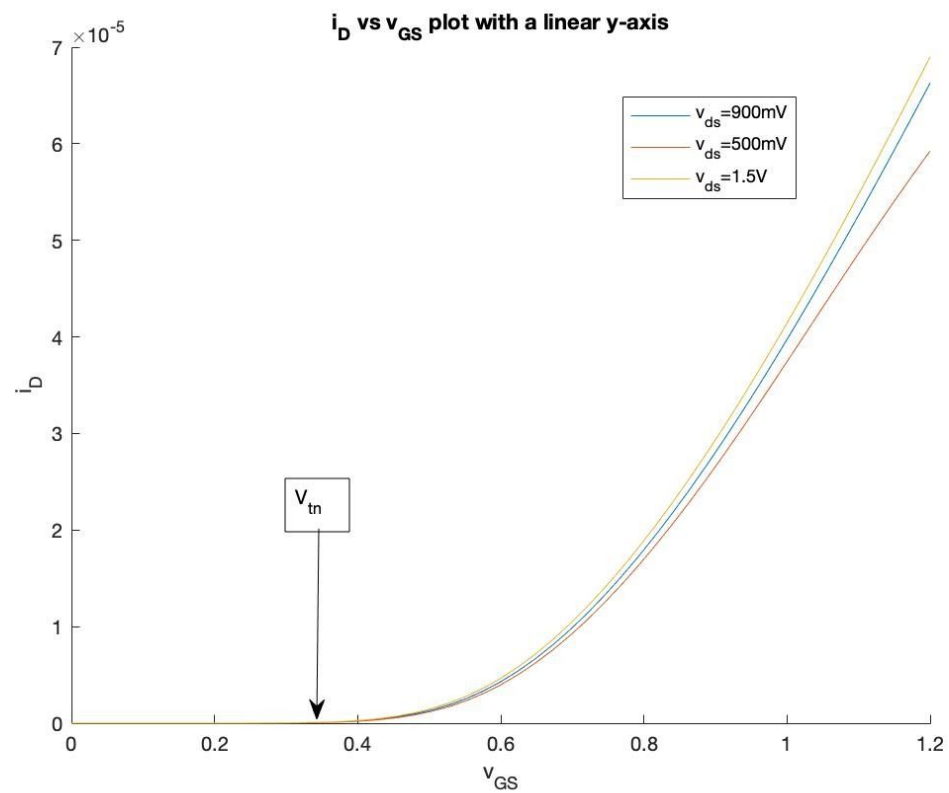


Figure 9: Linear plot for I_D vs V_{GS} with different values for V_{DS} .

We use the linear plot for I_D vs V_{GS} to estimate a value for the threshold voltage V_{tn} . We estimate that the $V_{tn} \approx 0.35\text{V}$.

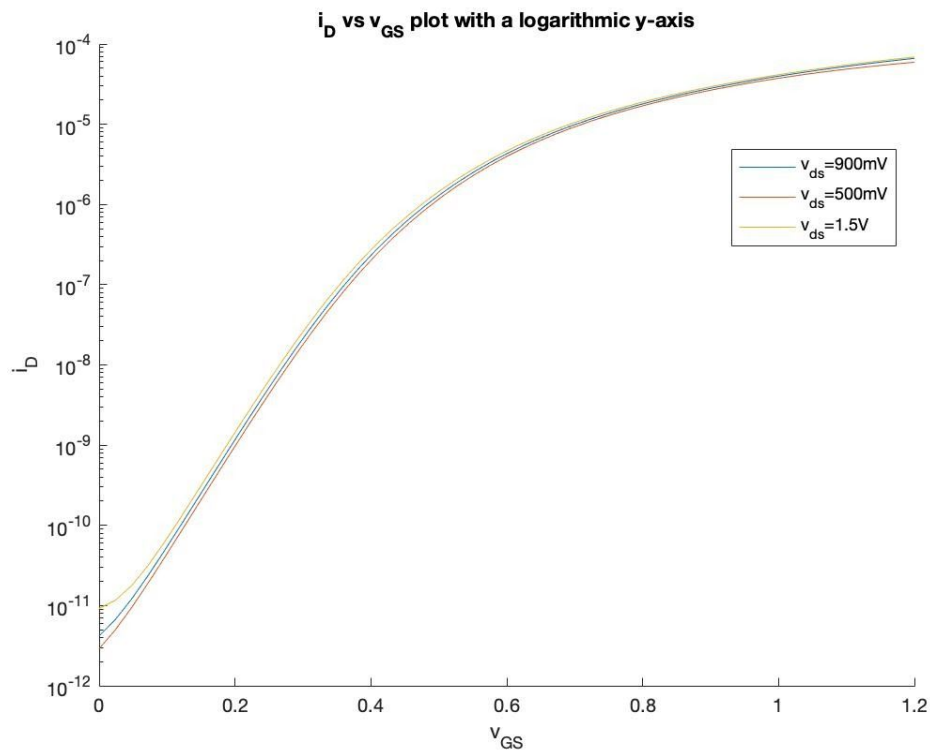


Figure 10: Logarithmic plot for I_D vs V_{GS} with different values for V_{DS} .

In the second part in this task we are going to present one plot with five different V_{GS1-5} in weak inversion and one plot with five different V_{GS6-10} in strong inversion. We should expect to see the saturation points for the plots in weak inversion to be around the same V_{DS} , whilst the saturation points for the plots in strong inversion should be at a higher V_{DS} as V_{GS} increases. The MATLAB code for plotting the I_D vs V_{GS} in strong and weak inversion:

Vgs plots:
1.33/2

```

29. %Plotting the id vs vds for different vgs in strong and weak inversion:
30. data300 = csvread('task1_part2_300.csv',1);
31. data275 = csvread('task1_part2_275.csv',1);
32. data250 = csvread('task1_part2_250.csv',1);
33. data225 = csvread('task1_part2_225.csv',1);
34. data200 = csvread('task1_part2_200.csv',1);
35.
36.
37. %Weak inversion
38. figure()
39. hold on;
40. plot(data300(:,1),data300(:,2))
41. plot(data275(:,1),data275(:,2))
42. plot(data250(:,1),data250(:,2))
43. plot(data225(:,1),data225(:,2))
44. plot(data200(:,1),data200(:,2))
45. title("i_{D} vs v_{DS} plots each curve with different v_{GS} in weak
    inversion")
46. xlabel("v_{DS}")
47. ylabel("i_{D}")
48. legend("v_{gs}=300mV", "v_{gs}=275mV", "v_{gs}=250mV", "v_{gs}=225mV",
    "v_{gs}=200mV")
49. hold off;
50.
51. data350 = csvread('task1_part2_350.csv',1); % Read the data
52. data450 = csvread('task1_part2_450.csv',1); % Read the data
53. data550 = csvread('task1_part2_550.csv',1); % Read the data
54. data650 = csvread('task1_part2_650.csv',1); % Read the data
55. data750 = csvread('task1_part2_750.csv',1); % Read the data
56.
57. %Strong inversion
58. figure()
59. hold on;
60. plot(data350(:,1),data350(:,2))
61. plot(data450(:,1),data450(:,2))
62. plot(data550(:,1),data550(:,2))
63. plot(data650(:,1),data650(:,2))
64. plot(data750(:,1),data750(:,2))
65. title("i_{D} vs v_{DS} plots each curve with different v_{GS} in strong
    inversion")
66. xlabel("v_{DS}")
67. ylabel("i_{D}")
68. legend("v_{gs}=350mV", "v_{gs}=450mV", "v_{gs}=550V", "v_{gs}=650",
    "v_{gs}=750")
69. hold off;

```

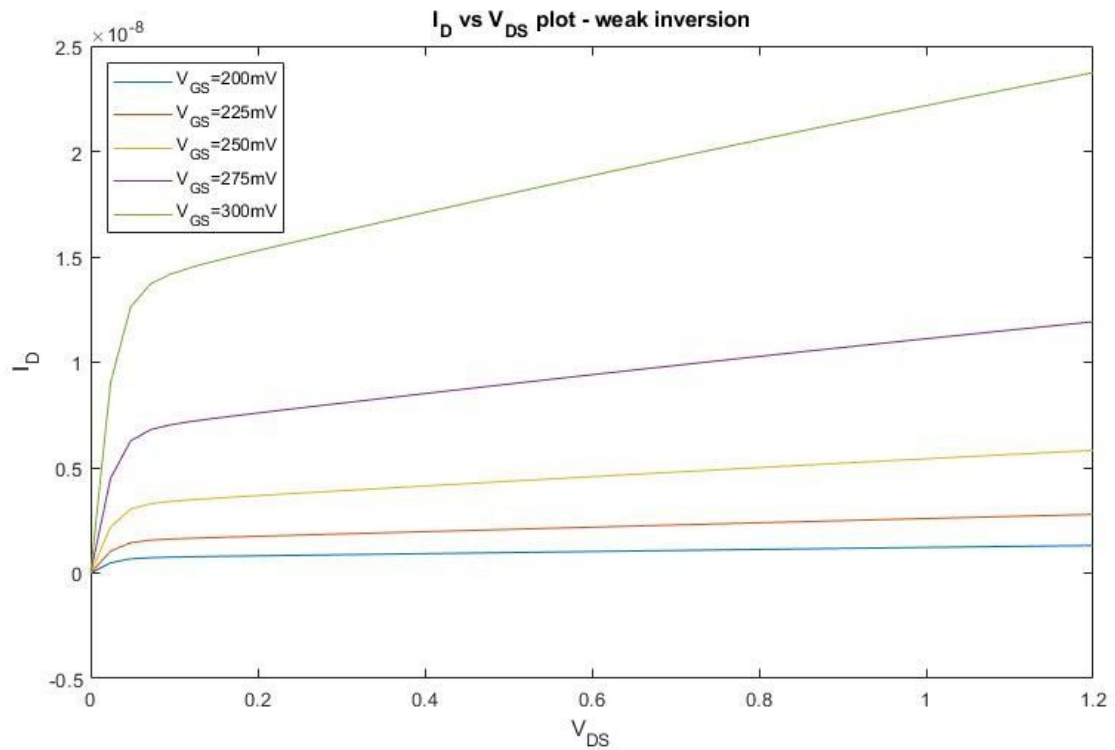


Figure 11: i_D vs v_{DS} plot with five different V_{GS} in weak inversion.

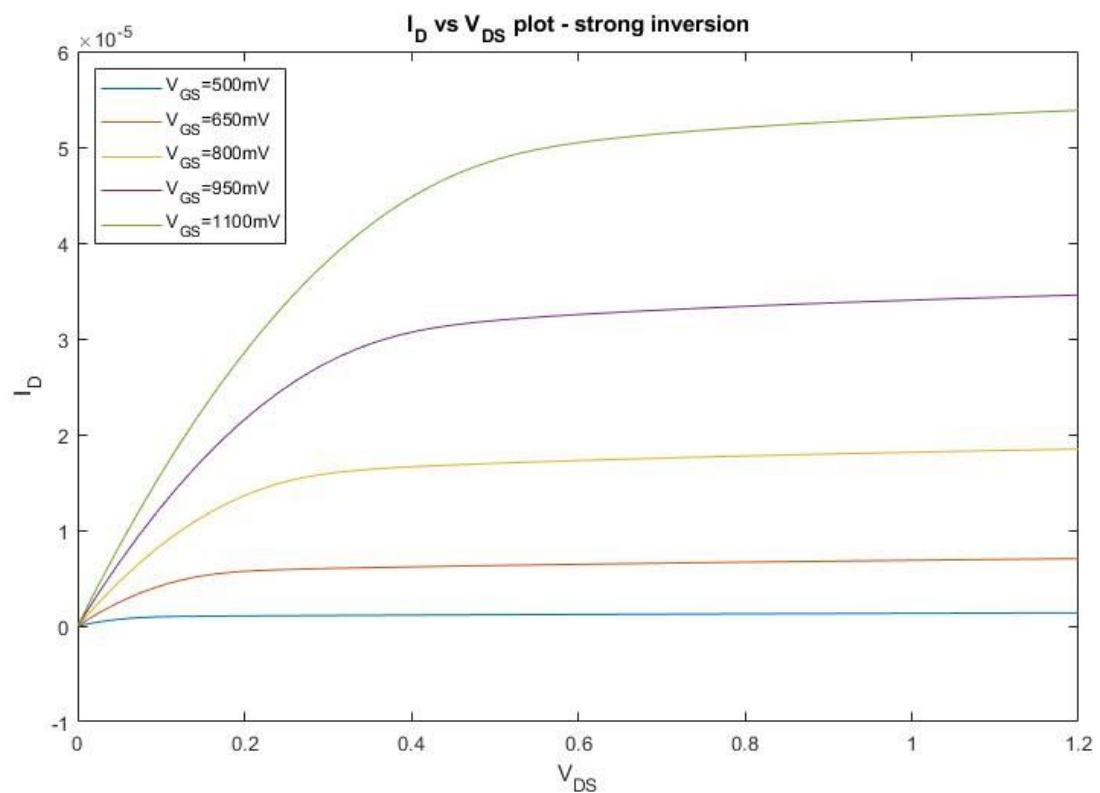


Figure 12: i_D vs v_{DS} plot with five different V_{GS} in strong inversion.

task 1: 3.33/4

Vds plots:2/2

The plots look as expected, with a clear difference in saturation points for the strong inversion plots, but similar saturation points for the weak inversion plots.

Task 2

In this task we are going to make a Matlab function that uses the full EKV-model. The function has the inputs parameters k_n , V_{tn} , λ , a constant v_{DS} and a sweep vector of v_{GS} values. The Matlab code for the EKV-model:

```

70. %EKV-model
71. function [id] = NFET(V_tn, kn,lambda, V_gs, V_ds)
72.     V_T = 0.026;
73.     V_s = 0;
74.     V_g = V_gs + V_s;
75.     V_d = V_ds + V_s;
76.     i_s = 2*kn.*(V_tn)^2;
77.     i_f = i_s.*log(1 + exp((V_g - V_tn - V_s)/(2.*V_T))).^2.*(1 +
    lambda.*V_ds);
78.     i_r = i_s.*log(1 + exp((V_g - V_tn - V_d)/(2.*V_T))).^2.*(1 +
    lambda.*V_ds);
79.     id = i_f - i_r;
80. end
81.
82. V_tn = 0.35;
83. V_gs = linspace(0,1.2, 100);
84. V_ds = 0.9;
85. kn = 1e-6;
86. lambda = 0.02;
87.
88. i = NFET(V_tn, kn,lambda, V_gs, V_ds)
89.
90. %Plotting the EKV-model
91. figure()
92. hold on
93. plot(V_gs, i, "--")
94. plot(data900(:,1),data900(:,2))
95. title("i_{D} vs v_{GS} plot with a linear y-axis")
96. xlabel("i_{GS}")
97. ylabel("i_{D}")
98. legend("EKV", "Simulation")
99. hold off;

```

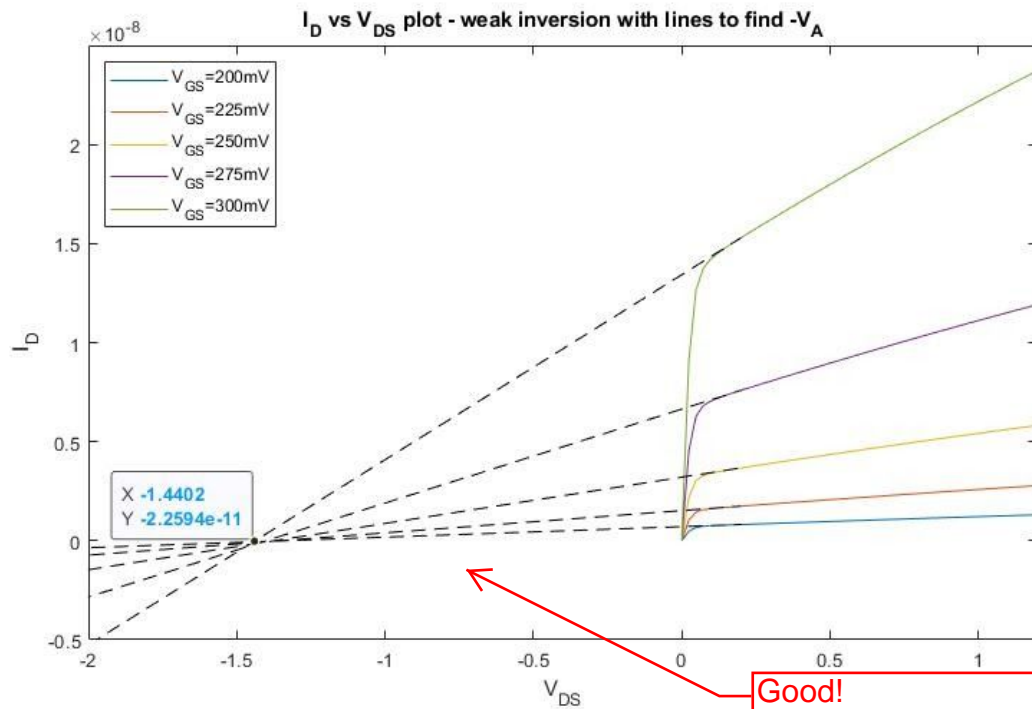


Figure 13: i_D vs v_{DS} plot with five different V_{GS} in weak inversion, with extended lines and marked crossing between the lines.

In Figure 13 we have plotted extended lines from the plots in weak inversion. We can see that these lines cross each other at about -1.44V. This gives us:

$$-V_A = -1.44V$$

From page 328 in Sedra Smith - Microelectronic Circuits, figure 5.17 we then get:

$$-V_A = -1/\lambda$$

So this is the value for the device parameter we get from the simulation:

$$\lambda = -1/-V_A = -1/-1.44V = 0.694V^{-1}$$

We tried using $\lambda = 0.694V^{-1}$ when plotting the EKV model, but the curve for the model did not match the curve for the simulation.

We think the reason for this is because the EKV model treats the device as being symmetrical. In our simulation we get a lambda that then doesn't fit the desired EKV model. The λ parameter is normally a small value and ranging from 0.005 to 0.02 V^{-1} . We tried different values in that range and got the best fit when we set the MOSFET device parameter to 0.02 V^{-1} .

Another parameter that we need to find a value so our EKV model fits the desired curve is the MOSFET transconductance k_n . This parameter can be tweaked by changing W/L . In our case we have $W = L = 540\text{ nm}$. What we did to find the right

k_n for the model: first find the right dimension for our curve and then the right value in that dimension. We opted for $k_n = 1 * 10^{-6} A/V^2$. We use the estimated V_{tn} we have from task 1 and we get the following plot in Figure 14:

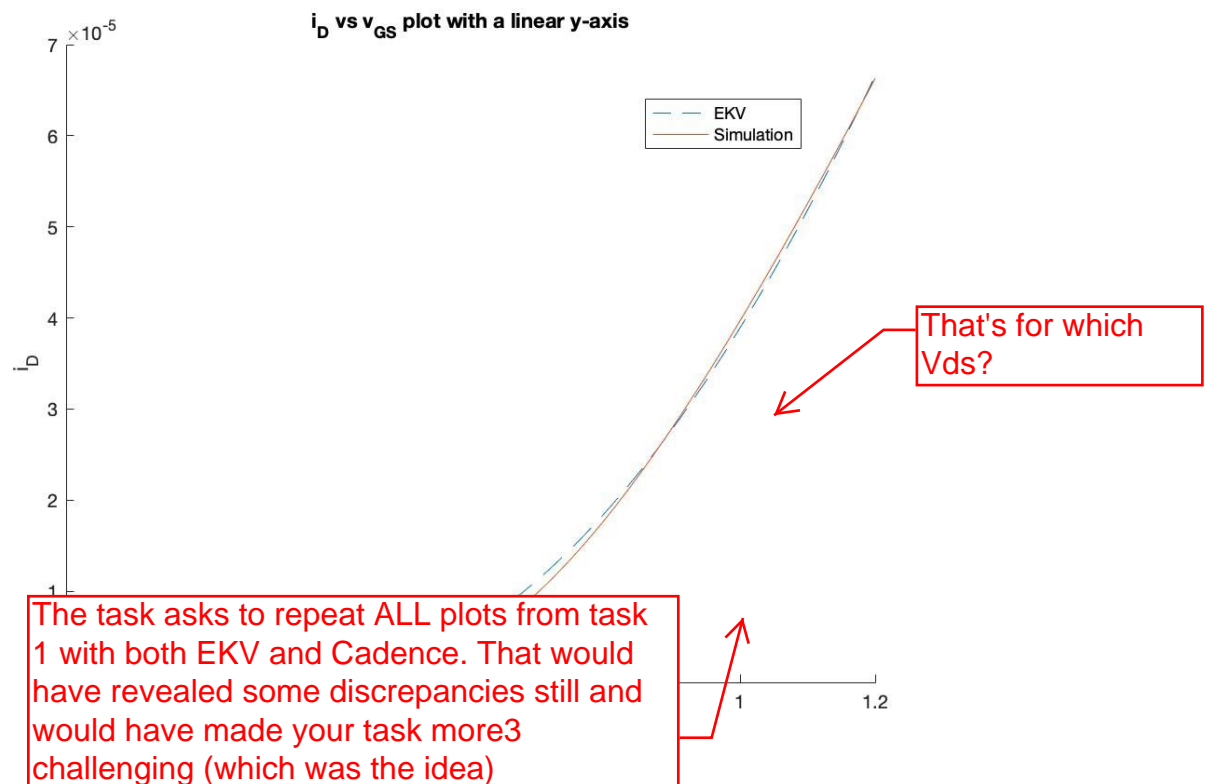


Figure 14: i_D vs v_{GS} plot comparing the EKV-model and simulation curve.

As we can see in Figure 14 the model fits very well with simulation. There are some differences between the EKV-model and the simulation curve, but small once. Right after the threshold voltage the EKV-model has a bigger increase than the simulation. This may be because of how the model treats the device as being symmetrical and does not make any difference between source and drain.

task 2:
 EKV: 1/1
 parameters: 1/1.5
 plots: 0.5/1.5
 total:
 2/4