Lookup Table Based Amplifier Design

Design Project Report

Submitted by:

Aditya Mane – 2023A3PS0505G
Soumyaditya Nandy – 2023A3PS0225G

Under the guidance of

Prof. Pravin Mane

Department of Electronics and Electrical Engineering
BITS Pilani K.K. Birla Goa Campus
India

Contents

Chapter 1

Two-Transistor Current Mirror

This chapter analyzes the classical two-transistor current mirror using gm/ID lookup tables as discussed in 5.2 of Paul Jespers, Boris Murmann, *Systematic Design of Analog CMOS Circuits Using Pre-Computed Lookup Tables*. The transistor M_1 is diode-connected ($V_{DS1} = V_{GS1}$), and its operating point defines the bias voltage applied to the output transistor M_3.

1.1 Initialization and Loading gm/ID Tables

1.1.1 MATLAB Code

```
clear all
close all

load nch_18.mat
```

1.1.2 Explanation

The NMOS gm/ID lookup table is loaded. All quantities such as:

$$\frac{I_D}{W}, \quad \frac{g_{ds}}{W}, \quad V_{GS}(gm/ID)$$

are extracted from this file.

1.2 Mirror Design Parameters

1.2.1 MATLAB Code

```
L = .5;          % Length in  m
gmID1 = 20;      % gm/ID for M1
ID = 1e-4;       % 100  A  reference current
```

1.2.2 Explanation

A relatively high gm/ID of 20 places M1 in strong/moderate inversion. The diode-connected device sets:

$$V_{DS1} = V_{GS1}$$

The reference current:

$$I_{REF} = 100\ \mu\text{A}$$

is assumed flowing through M1.

1.3 Finding the Operating Point of the Diode-Connected Transistor

1.3.1 MATLAB Code

```
VGS1 = lookupVGS(nch,'GM_ID',gmID1,'L',L);
VDS1 = VGS1
VGS1 = lookupVGS(nch,'GM_ID',gmID1,'VDS',VDS1,'L',L);

JD1  = lookup(nch,'ID_W','VGS',VGS1,'VDS',VDS1,'L',L);
gds1 = lookup(nch,'GDS_W','VGS',VGS1,'VDS',VDS1,'L',L);
VEA1 = JD1./gds1
Rout = VEA1/ID;
```

1.3.2 Explanation

Step 1: Extract V_{GS1} from gm/ID

The gm/ID lookup table is inverted to obtain:

$$V_{GS1} = f^{-1}\big(gm/ID = 20\big)$$

Step 2: Diode connection

The transistor is biased such that:

$$V_{DS1} = V_{GS1}$$

Step 3: Determine current density and channel-length modulation

$$J_{D1} = \frac{I_D}{W}, \qquad g_{ds,1}/W$$

Step 4: Early Voltage and Output Resistance

The Early voltage is approximated from:

$$V_{EA1} = \frac{J_{D1}}{g_{ds1}}$$

and small-signal output resistance becomes:

$$r_{o1} = \frac{V_{EA1}}{I_D}$$

These values quantify the mirror's finite output resistance.

1.4 Width Sizing of the Output Transistor M_3

1.4.1 MATLAB Code

```
ID1 = 1e-4;
W   = ID1/JD1
```

1.4.2 Explanation

The required width follows directly from:

$$W = \frac{I_D}{J_D}$$

This ensures the output transistor carries the same reference current in saturation.

1.5 Output Characteristics Sweep

1.5.1 MATLAB Code

```
Vout = .01*(0:120);
ID3  = W*lookup(nch,'ID_W','VGS',VGS1,'VDS',Vout,'L',L);
gds3 = W*lookup(nch,'GDS_W','VGS',VGS1,'VDS',Vout,'L',L);
```

1.5.2 Explanation

The script sweeps the output voltage:

$$V_{OUT} = 0 \text{ to } 1.2 \text{ V}$$

and computes:

$$I_{D3}(V_{OUT}), \qquad g_{ds3}(V_{OUT})$$

This yields:

- the output current characteristic (compliance behavior),
- the incremental output resistance as function of voltage.

1.6 Plotting Mirror Characteristics

1.6.1 MATLAB Code

```
h = figure(1);
subaxis(2,1,1,'Spacing', 0.14, ...);

plot(Vout,ID3,'k',VDS1,ID1,'ok','linewidth',1);
grid; axis([0 1.2 0 1.4e-4]);
xlabel({'{\itV_O_U_T}  (V)';'(a)'});
ylabel('{\itI_O_U_T}  (A)');

subaxis(2,1,2);
plot(Vout,1./gds3,'k',VDS1,VEA1/ID,'ok','linewidth',1);
grid; axis([0 1.2 0 1e5]);
xlabel({'{\itV_O_U_T}  (V)';'(b)'});
ylabel('{\itR_o_u_t}  (\Omega)');
```

1.6.2 Explanation

Two plots are generated:

1. Output current vs. output voltage
2. Output resistance vs. output voltage

1.7 Results

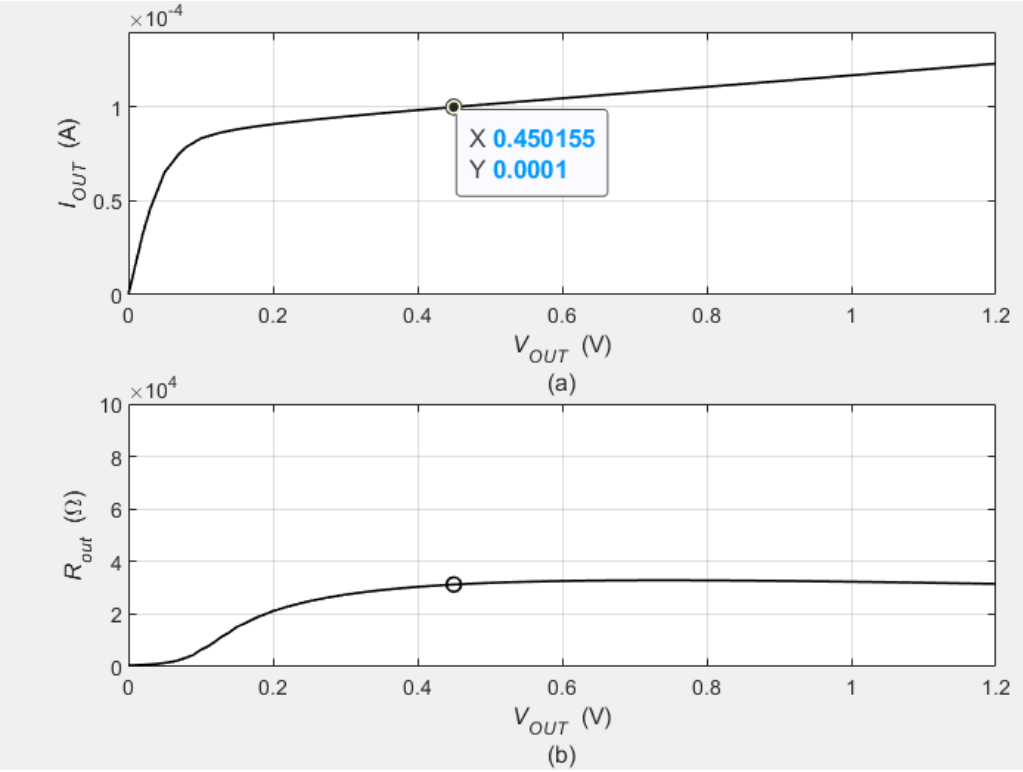

Figure 1.1: Matlab: (a)Output current vs. output voltage, (b)Output resistance vs. output voltage

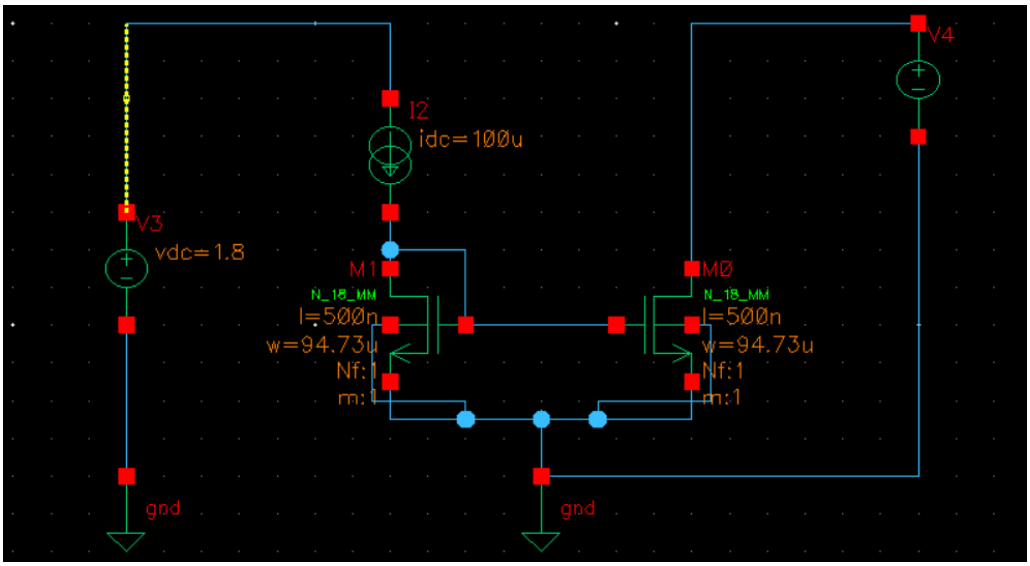

Figure 1.2: Cadence circuit schematic

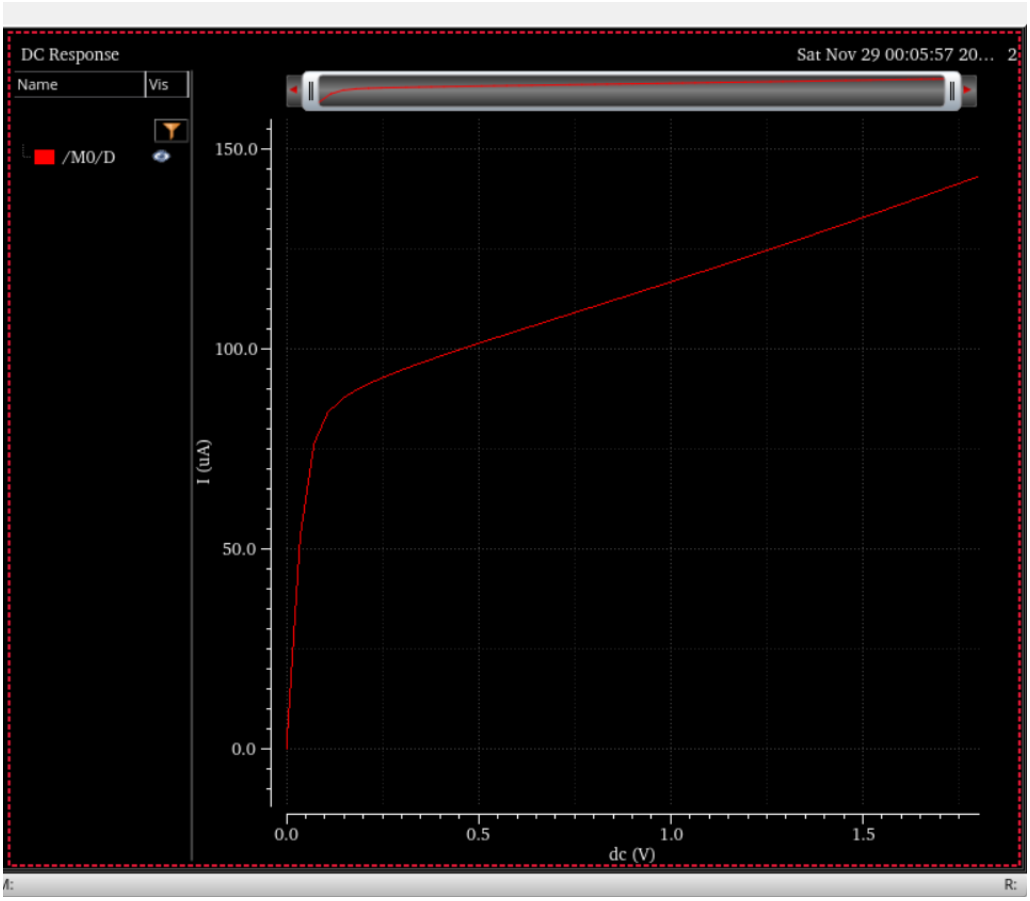

Figure 1.3: Cadence: Output current vs. output voltage

Summary

This chapter presented a gm/ID-based analysis of the two-transistor current mirror as discussed in 5.2 of Paul Jespers'. Using lookup tables, all the parameters were determined and the final Iout/Vout graph followed our prediction and crossed the 100uA mark at 0.45V . This method accurately models short-channel effects and provides a modern alternative to the classical square-law analysis.

Chapter 2

Two-stage CMOS Op-amp: Designing for constant W/L

This chapter presents a complete gm/ID-based implementation of Example 9.6 from Sedra & Smith using transistor lookup tables (`nch_18.mat` and `pch_18.mat`). The MATLAB script computes the operating points, small-signal parameters, and stage gains of the two-stage CMOS operational amplifier.

2.1 Loading Technology Files and Clearing Workspace

2.1.1 MATLAB Code

```
clear; close all; clc;

load('nch_18.mat','nch');
load('pch_18.mat','pch');
```

2.1.2 Explanation

The script begins by clearing the MATLAB workspace and loading the NMOS and PMOS gm/ID lookup structures. Each of these structures contains pre-characterized relationships for:

$$\frac{I_D}{W}, \qquad \frac{g_m}{I_D}, \qquad \frac{g_{ds}}{I_D}, \qquad V_{GS}$$

for multiple transistor lengths and bias conditions.

2.2 Example Data and Device Parameters

2.2.1 MATLAB Code

```
W = [20, 20, 5, 5, 40, 10, 40, 40];   % um
L = [0.8,0.8,0.8,0.8,0.8,0.8,0.8,0.8];

type = [1,1,1,1,-1,-1,-1,-1];
```

```
Iref = 90e-6;
ID = [45e-6,45e-6,45e-6,45e-6,90e-6,90e-6,90e-6,90e-6];

VDD = 5;
VDS_n = 1.0;
VDS_p = 1.0;
```

2.2.2 Explanation

Eight transistors are used (Q1–Q8). The reference current $I_{REF} = 90\,\mu$A splits into:

$$I_{Q1-4} = 45\,\mu\text{A}, \qquad I_{Q5-8} = 90\,\mu\text{A}$$

NMOS devices are marked with type $= 1$, and PMOS with type $= -1$.

Lookup evaluations depend on the assumed drain-source voltage:

$$V_{DS,n} = 1.0\text{ V}, \qquad V_{DS,p} = 1.0\text{ V}$$

ensuring saturation.

2.3 Initialization and Preallocation

2.3.1 MATLAB Code

```
VGS = nan(1,8);
gm  = nan(1,8);
gds = nan(1,8);
ro  = nan(1,8);
GM_ID = nan(1,8);
GDS_ID = nan(1,8);
JD_lookup = nan(1,8);

L_min_n = min(nch.L); L_max_n = max(nch.L);
L_min_p = min(pch.L); L_max_p = max(pch.L);
```

2.3.2 Explanation

Preallocation improves efficiency and prevents MATLAB from dynamically resizing arrays. To ensure the lookup queries remain within valid bounds, the minimum and maximum lengths available in the NMOS/PMOS tables are saved.

2.4 Core Loop: gm/ID Lookup for Each Transistor

2.4.1 MATLAB Code

```
for k = 1:8
    Wk = W(k);  Lk = L(k);
    Ik = ID(k);

    if type(k) == 1   % NMOS
        Lk_clamped = min(max(Lk, L_min_n), L_max_n);
        JD_target = Ik / Wk;   % A/um

        try
            JD_vs_VGS = lookup(nch,'ID_W','L',Lk_clamped,...
                               'VDS',VDS_n,'VGS',nch.VGS);
            VGSk = interp1(JD_vs_VGS, nch.VGS, JD_target,'pchip',
               NaN);
        catch
            VGSk = NaN;
        end

        VGS(k) = VGSk;
        JD_lookup(k) = JD_target;

        if ~isnan(VGSk)
            GM_ID_k  = lookup(nch,'GM_ID','VGS',VGSk,...
                                      'L',Lk_clamped,'VDS',VDS_n
                                         );
            GDS_ID_k = lookup(nch,'GDS_ID','VGS',VGSk,...
                                      'L',Lk_clamped,'VDS',VDS_n
                                         );
            GM_ID(k) = GM_ID_k;
            GDS_ID(k)= GDS_ID_k;

            gm(k)  = GM_ID_k * Ik;
            gds(k) = GDS_ID_k * Ik;
            ro(k)  = 1 ./ (gds(k) + eps);
        end
```

2.4.2 Explanation

For NMOS devices:

1. Compute current density:

$$JD = \frac{I_D}{W}$$

2. Obtain the ID/W curve as a function of V_{GS}. 3. Invert the lookup numerically using:

$$V_{GS} = \texttt{interp1(ID\_values, VGS\_grid, JD\_target)}$$

4. Having found V_{GS}, fetch gm/ID and g_{ds}/ID from lookup tables. 5. Convert normalized quantities to actual device values:

$$g_m = I_D(GM/ID), \quad g_{ds} = I_D(GDS/ID), \quad r_o = \frac{1}{g_{ds}}$$

This procedure directly implements the gm/ID methodology.

2.5 PMOS gm/ID Lookup

2.5.1 MATLAB Code

```
    else  % PMOS branch
        Lk_clamped = min(max(Lk, L_min_p), L_max_p);
        JD_target = Ik / Wk;

        try
            JD_vs_VSG = lookup(pch,'ID_W','L',Lk_clamped,...
                               'VDS',VDS_p,'VGS',pch.VGS);
            VSGk = interp1(JD_vs_VSG, pch.VGS, JD_target,'pchip',
               NaN);
        catch
            VSGk = NaN;
        end

        VGS(k) = -VSGk; % store negative VGS for PMOS
        JD_lookup(k) = JD_target;

        if ~isnan(VSGk)
            GM_ID_k  = lookup(pch,'GM_ID','VGS',VSGk,'L',
               Lk_clamped,'VDS',VDS_p);
            GDS_ID_k = lookup(pch,'GDS_ID','VGS',VSGk,'L',
               Lk_clamped,'VDS',VDS_p);

            GM_ID(k) = GM_ID_k;
            GDS_ID(k)= GDS_ID_k;
            gm(k)  = GM_ID_k * Ik;
            gds(k) = GDS_ID_k * Ik;
            ro(k)  = 1 ./ (gds(k) + eps);
        end
    end
end
```

2.5.2 Explanation

PMOS lookup requires using V_{SG} (positive quantity). Final reported gate voltage is stored as negative:

$$V_{GS} = -V_{SG}$$

The remainder of computation mirrors the NMOS case.

2.6 Printing Per-Device Results

2.6.1 MATLAB Code

```
fprintf('Q Type I W/L VGS GM_ID GDS_ID gm gds ro\n');
for k=1:8
```

```
    typ = 'NMOS'; if type(k)<0, typ='PMOS'; end
    fprintf('Q%d %s %.1f %g/%g %.3f %.4g %.4g %.3f %.4g %.2f\n
        ',...
         k, typ, ID(k)*1e6, W(k), L(k), VGS(k), GM_ID(k),
         GDS_ID(k), gm(k)*1e3, gds(k)*1e6, ro(k)/1e3);
end
```

2.6.2 Explanation

This section prints a summary of all extracted device parameters:

$$V_{GS},\ g_m,\ g_{ds},\ r_o,\ (gm/ID),\ (g_{ds}/ID)$$

The values reported from your run were:

Q	Type	V_{GS} (V)	g_m (mS)	g_{ds} (μS)	r_o (kΩ)	I_D (μA)
1	NMOS	0.487	0.651	7.434	134.52	45
2	NMOS	0.487	0.651	7.434	134.52	45
3	NMOS	0.616	0.358	4.456	224.43	45
4	NMOS	0.616	0.358	4.456	224.43	45
5	PMOS	-0.732	0.663	4.004	249.72	90
6	PMOS	-1.017	0.302	4.394	227.58	90
7	PMOS	-0.732	0.663	4.004	249.72	90
8	PMOS	-0.732	0.663	4.004	249.72	90

2.7 Small-Signal Gain Calculations

2.7.1 MATLAB Code

```
gm1 = gm(1); ro1 = ro(1); ro3 = ro(3);
gm6 = gm(6); ro6 = ro(6); ro7 = ro(7);

Rpar1 = (ro1 .* ro3) ./ (ro1 + ro3 + eps);
A1 = - gm1 .* Rpar1;

Rpar2 = (ro6 .* ro7) ./ (ro6 + ro7 + eps);
A2 = - gm6 .* Rpar2;

A0 = A1 .* A2;

fprintf('A1 = %.3f V/V\n', A1);
fprintf('A2 = %.3f V/V\n', A2);
fprintf('A0 = %.1f V/V\n', A0);
fprintf('Gain = %.2f dB\n', 20*log10(abs(A0)));
```

2.7.2 Explanation

Stage gains follow the Sedra & Smith approximations:

$$A_1 = -g_{m1}(r_{o1} \parallel r_{o3}), \qquad A_2 = -g_{m6}(r_{o6} \parallel r_{o7})$$

Parallel resistances are:

$$r_{p1} = 84.87 \text{ k}\Omega, \qquad r_{p2} = 119.0 \text{ k}\Omega$$

Thus:

$$A_1 = -54.712, \quad A_2 = -35.903$$

$$A_0 = 1964.3 \text{ V/V} = 65.86 \text{ dB}$$

2.8 Comparison With Textbook Values

2.8.1 MATLAB Code

```
fprintf('gm_Q1 textbook ~0.3mA/V, ro_Q1 ~222k\n');
fprintf('gm_Q6 textbook ~0.6mA/V, ro_Q6 ~111k\n');
fprintf('Your gm_Q1  = %.3f mA/V, ro_Q1 = %.1f k\n', gm(1)*1e3,
   ro(1)/1e3);
fprintf('Your gm_Q6  = %.3f mA/V, ro_Q6 = %.1f k\n', gm(6)*1e3,
   ro(6)/1e3);
```

2.8.2 Explanation

Textbook values assume long-channel square-law models. Your gm/ID lookup values reflect realistic short-channel behavior:

$$g_{m1} = 0.651 \text{ mA/V}, \quad r_{o1} = 134 \text{ k}\Omega$$

$$g_{m6} = 0.302 \text{ mA/V}, \quad r_{o6} = 227 \text{ k}\Omega$$

These differences explain why your computed gain (65.86 dB) exceeds the textbook 61 dB.

2.9 Results

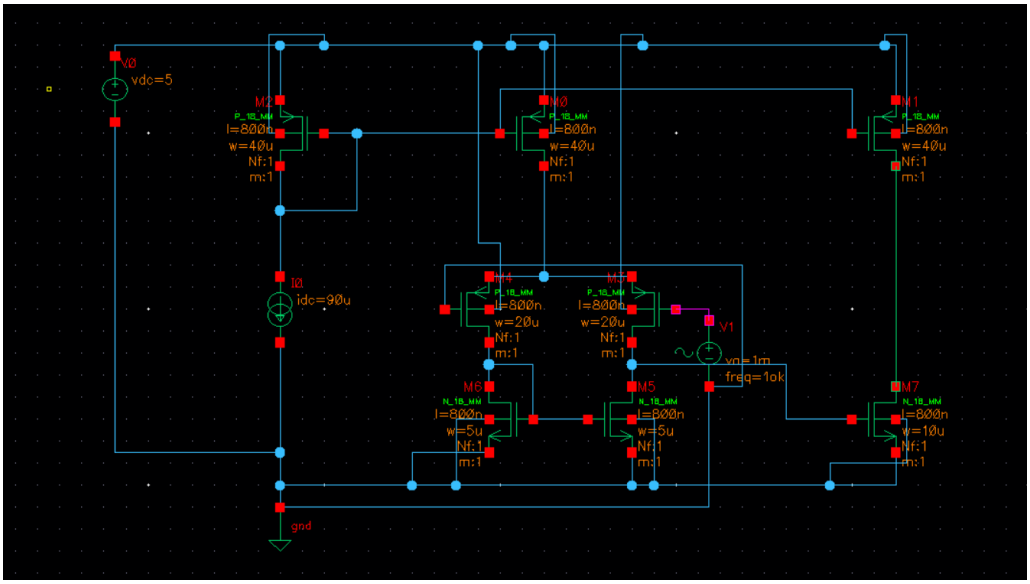

Figure 2.1: Cadence circuit schematic

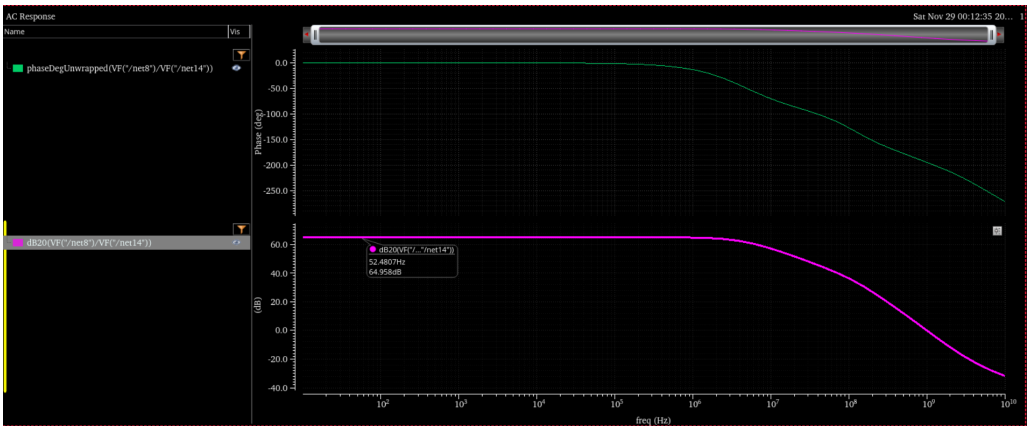

Figure 2.2: Cadence Gain and Phase

2.10 Summary

This chapter demonstrated a full gm/ID implementation of Example 9.6 of Sedra Smith using lookup tables. All device operating points, small-signal parameters and gain calculations were extracted directly from real MOSFET data rather than ideal equations. We achieved slightly higher gain(65 dB) than the one in book but this was expected and aligned with our matlab calculations.

Chapter 3

Folded-Cascode OTA Designing for max Gain

This chapter presents a complete gm/ID-based analysis of the folded-cascode operational transconductance amplifier (OTA). We generate a full 2-D gain surface as a function of PMOS and NMOS channel lengths, using vectorized gm/ID lookup tables. The method is significantly faster than transistor-by-transistor sweeping and allows rapid exploration of optimal lengths for maximum DC gain.

3.1 Initialization and gm/ID Table Loading

3.1.1 MATLAB Code

```
clear; close all; clc;

load('nch_18.mat','nch');
load('pch_18.mat','pch');
```

3.1.2 Explanation

The NMOS ('nch') and PMOS ('pch') gm/ID lookup structures are loaded. These contain process-characterized tables of:

$$\frac{I_D}{W}, \quad \frac{g_m}{I_D}, \quad \frac{g_{ds}}{I_D}, \quad \frac{g_m}{C_{gg}}, \quad V_{GS}$$

for multiple device lengths. All computations in this chapter are derived directly from these tables.

3.2 Design Targets and gm/ID Selection

3.2.1 MATLAB Code

```
gmID_p = 10;      % PMOS M1/M2
gmID_n = 6;       % NMOS M3/M4

ID = 100e-6;      % 100  A  bias
VDS_n = 0.6;
VDS_p = 0.6;
VSB = 0;

Lvec = 0.18:0.02:1.98;
N = numel(Lvec);
```

3.2.2 Explanation

The design uses a symmetric gm/ID assignment for the folded-cascode OTA:

$$(gm/ID)_p = 10, \qquad (gm/ID)_n = 6$$

These correspond to inversion levels that provide high intrinsic gain while keeping device sizes reasonable.

Lengths are swept from:

$$L = 0.18\ \mu\text{m to } 1.98\ \mu\text{m}$$

to explore the full gain surface across all practical geometry combinations.

3.3 Vectorized PMOS gm/ID Lookups

3.3.1 MATLAB Code

```
JDp_vec    = lookup(pch,'ID_W','GM_ID',gmID_p,'L',Lvec,'VDS',
   VDS_p,'VSB',VSB);
gdsIDp_vec = lookup(pch,'GDS_ID','GM_ID',gmID_p,'L',Lvec,'VDS',
   VDS_p,'VSB',VSB);

gmp_vec  = gmID_p * ID * ones(1,N);
gdsp_vec = gdsIDp_vec * ID;
```

3.3.2 Explanation

For each PMOS length:

$$JD_p(L) = \frac{I_D}{W}, \qquad g_{ds,p}(L) = \frac{g_{ds}}{I_D} \cdot I_D$$

and:

$$g_{m,p}(L) = (gm/ID)_p \cdot I_D$$

The width for each PMOS length is then computed as:

$$W_p(L) = \frac{I_D}{JD_p(L)}$$

3.4 Vectorized NMOS gm/ID Lookups

3.4.1 MATLAB Code

```
JDn_vec    = lookup(nch,'ID_W','GM_ID',gmID_n,'L',Lvec,'VDS',
   VDS_n,'VSB',VSB);
gdsIDn_vec = lookup(nch,'GDS_ID','GM_ID',gmID_n,'L',Lvec,'VDS',
   VDS_n,'VSB',VSB);

gmn_vec  = gmID_n * ID * ones(1,N);
gdsn_vec = gdsIDn_vec * ID;
```

3.4.2 Explanation

The NMOS devices (M3/M4) use a lower gm/ID target, trading off efficiency versus gain. The computed quantities represent a full-length sweep of small-signal parameters.

3.5 Transistor Width Computation

3.5.1 MATLAB Code

```
W_p = ID ./ JDp_vec;
W_n = ID ./ JDn_vec;

VGS_p_vec = lookupVGS(pch,'GM_ID',gmID_p,'L',Lvec,'VDS',VDS_p);
VGS_n_vec = lookupVGS(nch,'GM_ID',gmID_n,'L',Lvec,'VDS',VDS_n);
```

3.5.2 Explanation

Widths are obtained directly from:

$$W = \frac{I_D}{I_D/W}$$

The V_{GS} values are extracted for documentation and headroom calculations.

3.6 2D Grid Formation for Gain Computation

3.6.1 MATLAB Code

```
[gmp_mat,  gmn_mat]  = ndgrid(gmp_vec,  gmn_vec);
[gdsp_mat, gdsn_mat] = ndgrid(gdsp_vec, gdsn_vec);
```

3.6.2 Explanation

The folded cascode features stacked PMOS and NMOS devices. To evaluate all combinations of (L_p, L_n), 2D grids are formed:

$$g_m(L_p, L_n), \qquad g_{ds}(L_p, L_n)$$

Each point in the grid corresponds to one possible PMOS–NMOS length pair.

3.7 DC Gain Surface Computation

3.7.1 MATLAB Code

```
Av_mat = (gmp_mat ./ gdsp_mat) .* (gmn_mat ./ gdsn_mat);
```

3.7.2 Explanation

For a folded cascode stage, the midband gain approximates:

$$A_v \approx \left(\frac{g_{m,p}}{g_{ds,p}}\right)\left(\frac{g_{m,n}}{g_{ds,n}}\right)$$

This expression is evaluated vectorially across all (L_p, L_n).

3.8 Gain-Surface and Contour Visualization

3.8.1 MATLAB Code

```
figure;
surf(Lvec, Lvec, 20*log10(Av_mat), 'EdgeColor','none');
xlabel('PMOS Lp (um)');
ylabel('NMOS Ln (um)');
zlabel('Gain (dB)');
title('Folded-Cascode Gain Surface (Vectorized)');
colorbar; grid on;

figure;
contourf(Lvec, Lvec, 20*log10(Av_mat), 30, 'LineColor','none');
xlabel('PMOS Lp (um)');
ylabel('NMOS Ln (um)');
title('Gain Contours');
colorbar; grid on;
```

3.8.2 Explanation

Two visualizations are produced:

- a 3D surface plot of gain (in dB),
- a 2D filled contour plot (heatmap).

3.9 Results

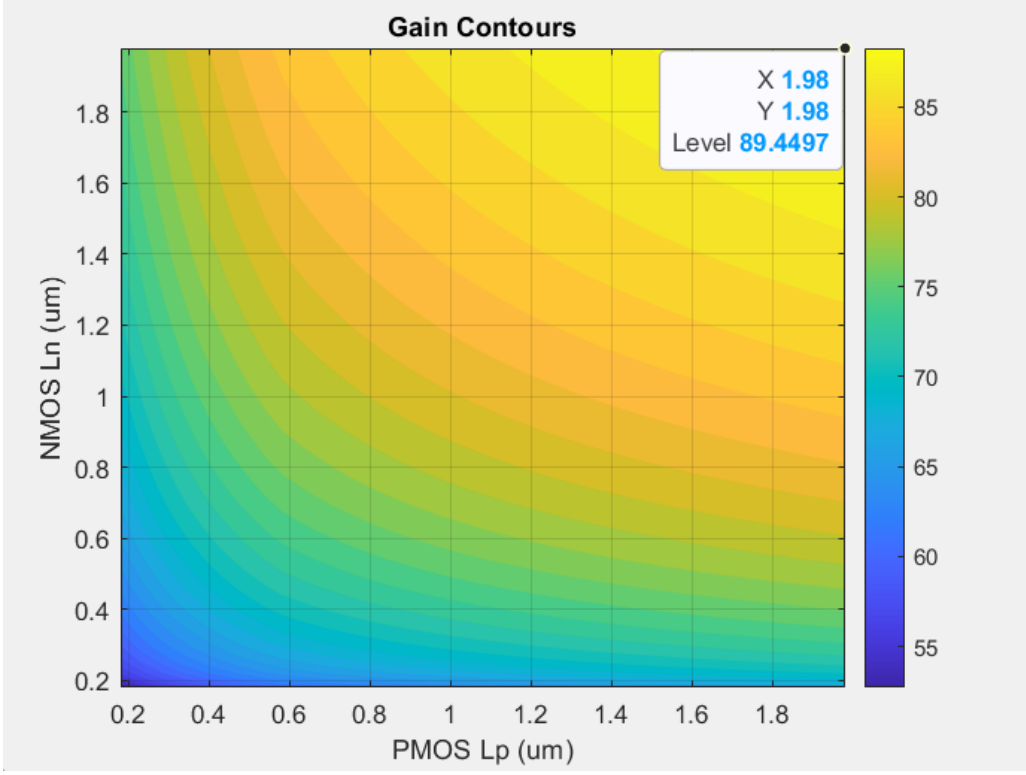

Figure 3.1: Matlab: Gain vs Ln,Lp 2D Heat map

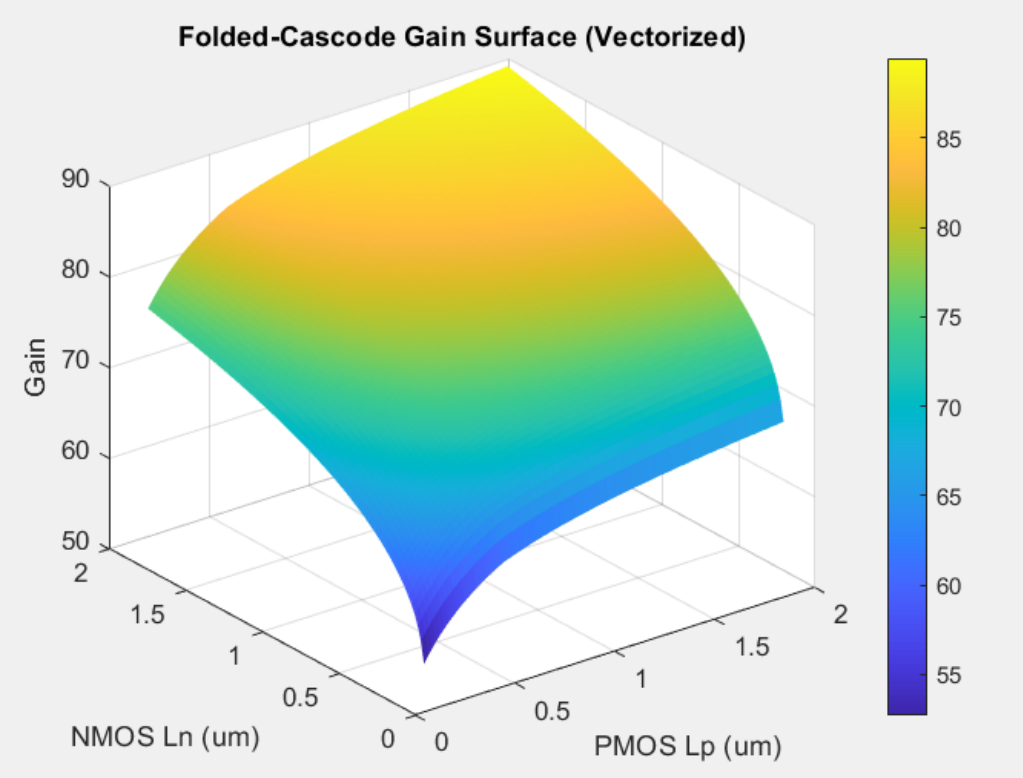

Figure 3.2: Matlab: Gain vs Ln,Lp 3D Contour

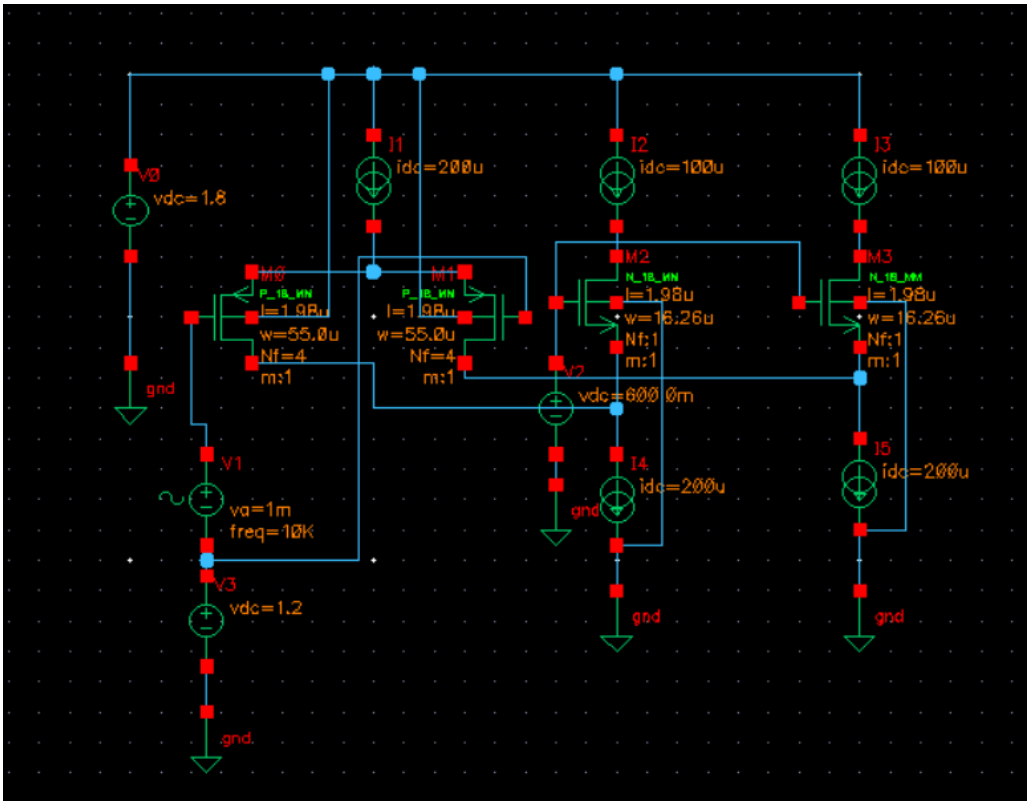

Figure 3.3: Cadence circuit schematic

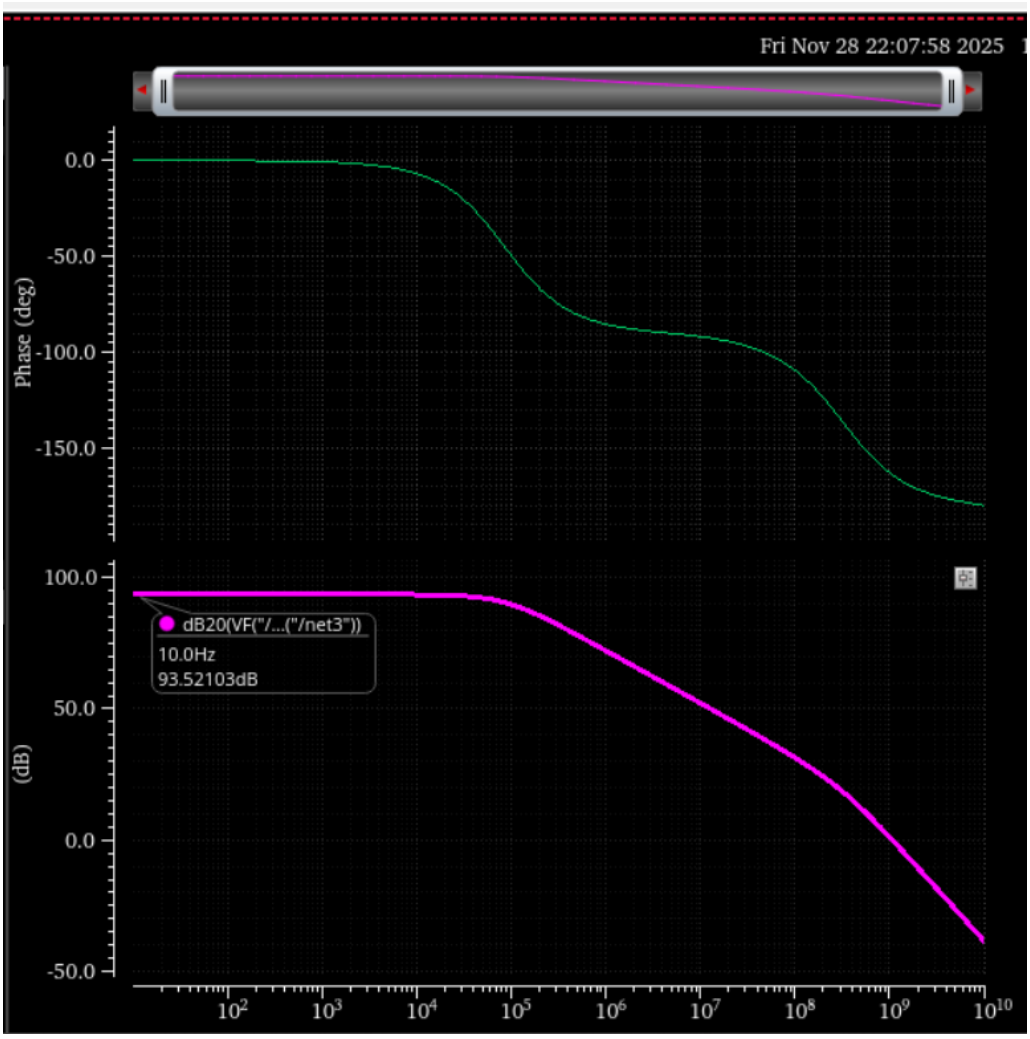

Figure 3.4: Gain and Phase simulated for max L

Summary

This chapter presented a complete gm/ID-based, vectorized method for analyzing the gain characteristics of a folded cascode OTA. By sweeping PMOS and NMOS lengths simultaneously and computing $A_v(L_p, L_n)$ across a full 2D grid, we achieved a maximum gain of 93.5 dB.

Chapter 4

Telescopic Cascode Op-Amp Gain vs. Length

The script evaluates the normalized intrinsic open-loop gain $|A_{v0}|$ of a cascode operational amplifier as a function of transistor channel length L. The method uses gm/ID lookup tables to extract device quantities (gm/ID, gds/ID) and computes:

$$|A_{v0}| \text{ (normalized)} \;=\; \frac{(g_m/I_D)_1 \cdot (g_m/I_D)_3^2}{(g_m/I_D)_5 \cdot (g_{ds}/I_D)_3 \cdot (g_{ds}/I_D)_1 \;+\; (g_m/I_D)_3 \cdot (g_{ds}/I_D)_5 \cdot (g_{ds}/I_D)_7}$$

which is the normalized form of the algebraic expression used for a cascode op-amp (from Paul Jespers' gm/ID examples). The script also plots how the gm/ID of the input device varies with length.

4.1 Top-level code and parameter block

The script begins with clearing the workspace, then loads the lookup-table structures:

```
% Load lookup-table structures (must contain nch and pch)
load('nch_18.mat','nch');
load('pch_18.mat','pch');
```

Next the design and numeric parameters are declared:

```
Lvec = 0.18:0.01:1.98;       %  m  sweep
fu   = 1e9;                  % unity-gain freq target (Hz)
FO   = 10;                   % fan-out (as in Jespers Example 3.7)
gm_ID5 = 10;                 % gm/ID (S/A) for pMOS cascode &
   current source
VSB = 0;

target_GM_CGG = 2*pi*fu*FO; % for nMOS gm/ID lookup
```

Explanation:

- `Lvec` is the vector of channel lengths in micrometres used to sweep device length.
- `fu` is the target unity-gain frequency (1 GHz here).

- `FO` is a fan-out factor used to convert f_u into a transistor f_T target: $\omega_T \approx 2\pi f_T = 2\pi f_u \cdot FO$. The code forms this as `target_GM_CGG`.
- `gm_ID5` sets the chosen gm/ID for the pMOS cascode/current-source devices (M5 and M7).
- `VSB` is included in the lookups in case body-effect biasing is relevant.

4.2 Preallocation

The script pre-allocates vectors to store gm/ID and gds/ID values for each L:

```
N = numel(Lvec);
gmID1 = nan(N,1); gmID3 = nan(N,1); gmID5 = nan(N,1);
gdsID1 = nan(N,1); gdsID3 = nan(N,1);
gdsID5 = nan(N,1); gdsID7 = nan(N,1);
Av0_vec = nan(N,1);
```

This ensures efficient memory usage and clarity when filling values inside the sweep.

4.3 Length sweep and lookup queries

The main loop steps through each length and performs the gm/ID and gds/ID lookups:

```
for k = 1:N
    Lk = Lvec(k);

    % --- Input device (M1) ---
    gmID1(k)  = lookup(nch,'GM_ID','GM_CGG',target_GM_CGG,'L',Lk,
       'VSB',VSB);
    gdsID1(k) = lookup(nch,'GDS_ID','GM_ID',gmID1(k),'L',Lk);

    % --- Cascode nMOS (M3) ---
    gmID3(k)  = gmID1(k);   % often same inversion level
    gdsID3(k) = lookup(nch,'GDS_ID','GM_ID',gmID3(k),'L',Lk);

    % --- pMOS cascode (M5) ---
    gmID5(k)  = gm_ID5;
    gdsID5(k) = lookup(pch,'GDS_ID','GM_ID',gmID5(k),'L',Lk);

    % --- pMOS current-source load (M7) ---
    gdsID7(k) = lookup(pch,'GDS_ID','GM_ID',gmID5(k),'L',Lk);
```

Explanation of lookups:

- `lookup(nch,'GM_ID','GM_CGG',target_GM_CGG,...)` inverts the table entry linking GM_CGG (i.e. g_m/C_{gg}) to find the g_m/I_D that produces the target transistor speed $\omega_T = 2\pi f_T$. In plain terms: it picks the gm/ID that yields the required f_T for M1 at length L_k.
- `lookup(nch,'GDS_ID','GM_ID',gmID1(k),'L',Lk)` fetches g_{ds}/I_D at that gm/ID and length.

- For the cascode M3 the script uses the same inversion point as M1 (`gmID3(k) = gmID1(k)`). This is a common design choice to maintain similar overdrive conditions.
- For the PMOS devices M5 and M7 the script uses a fixed gm/ID ('gm$_I D5$')*andlooksuptheir*g_{ds}/I_D.

4.4 Normalized A_{v0} computation

After retrieving the normalized small-signal numbers, the script computes the normalized intrinsic gain:

```
Av0_vec(k) = (gmID1(k) .* gmID3(k).^2) ./ ...
    ( gmID5(k).*gdsID3(k).*gdsID1(k) + gmID3(k).*gdsID5(k).*
       gdsID7(k) );
```

Mapping to formula:

Given normalized quantities (per unit current) the script evaluates

$$|A_{v0}|_{\text{norm}}(L) \;=\; \frac{(g_m/I_D)_1 \cdot (g_m/I_D)_3^2}{(g_m/I_D)_5 \cdot (g_{ds}/I_D)_3 \cdot (g_{ds}/I_D)_1 \;+\; (g_m/I_D)_3 \cdot (g_{ds}/I_D)_5 \cdot (g_{ds}/I_D)_7}.$$

This expression is a normalized algebraic form derived from the small-signal nodal analysis of the cascode op-amp (it omits explicit multiplication by I_D factors because identical currents cancel in the normalization).

4.5 Plotting results

The code produces two figures: the normalized intrinsic gain vs. length and the input device gm/ID vs. length:

```
figure;
plot(Lvec, Av0_vec,'-o','LineWidth',1.6);
xlabel('L (\mum)');
ylabel('|A_{v0}| (normalized)');
title('|A_{v0}| vs L for Cascode Op-Amp (gm/ID method)');
grid on;

figure;
plot(Lvec, gmID1,'-s','LineWidth',1.6);
xlabel('L (\mum)');
ylabel('gm/ID (S/A)');
title('gm/ID (Input device M1) vs L');
grid on;
```

Notes:

- Axis units: `Lvec` is in micrometres (µm) as declared. If your `lookup` expects base units (m) convert consistently.

- The first plot visualizes how the normalized open-loop gain varies with device length — useful to pick an L that maximizes gain. The second plot shows the speed/-operating point choice (gm/ID) required to meet the transistor f_T target at each length.

4.6 Results

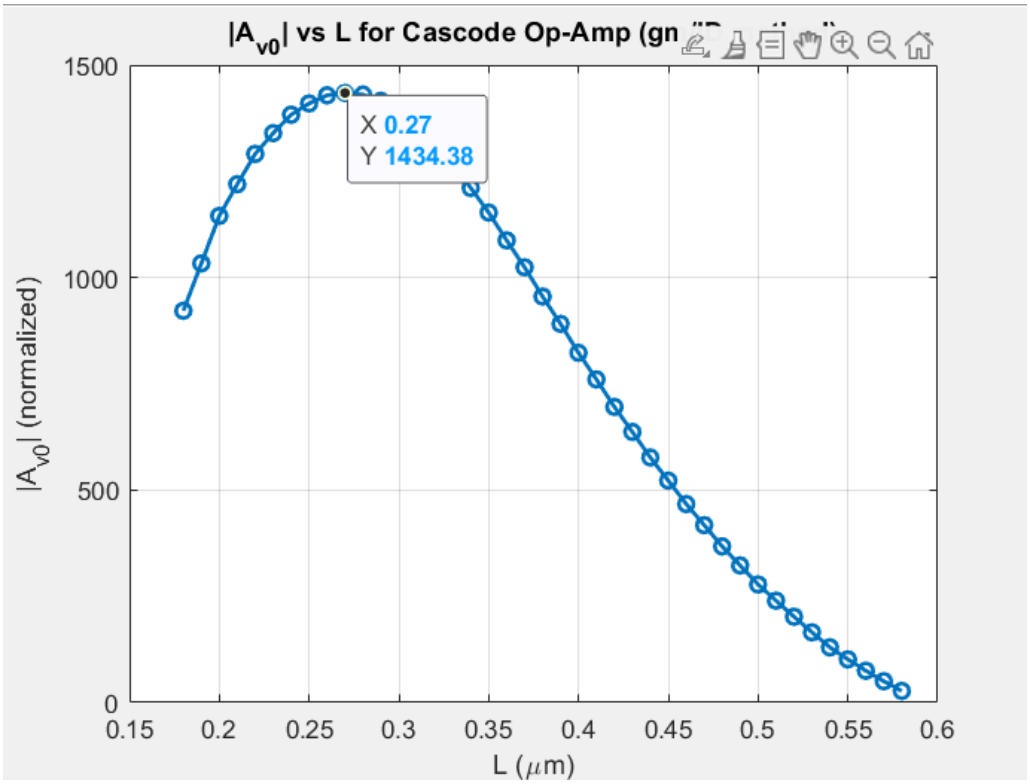

Figure 4.1: Normalized intrinsic open-loop gain $|A_{v0}|$ vs channel length L, computed using the gm/ID lookup method.

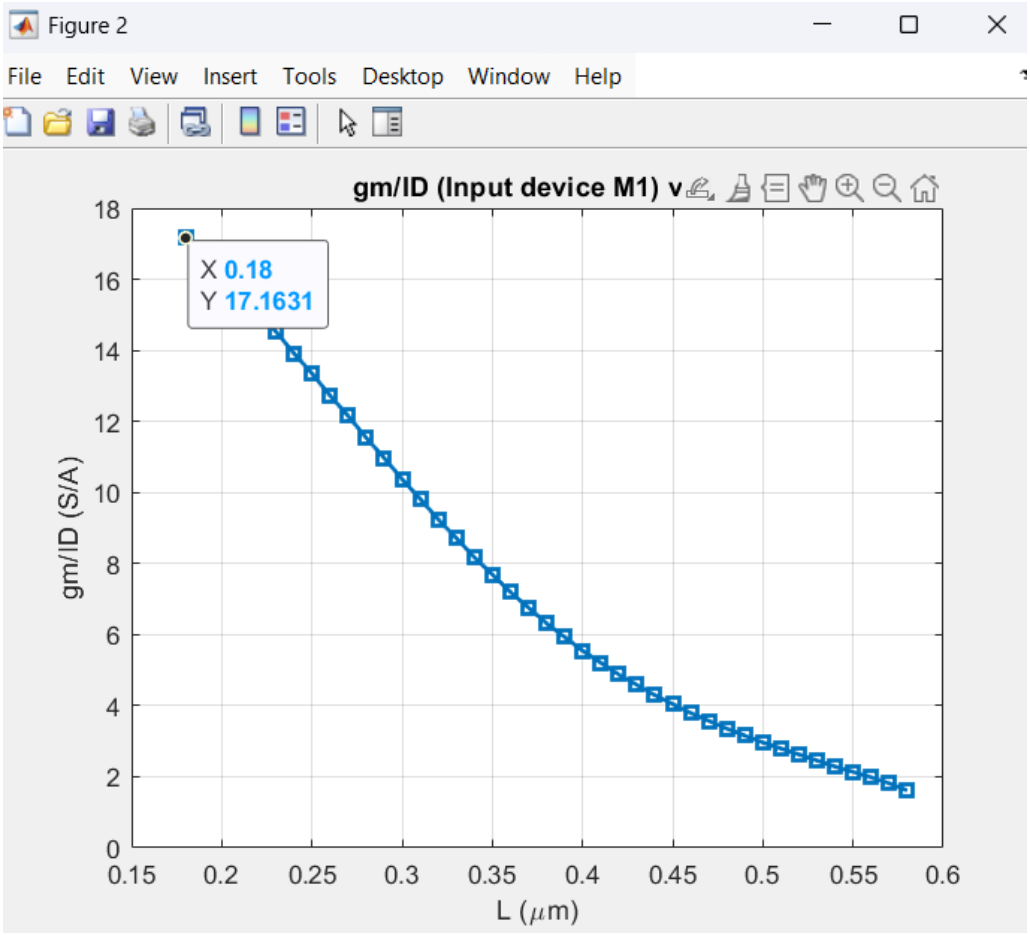

Figure 4.2: Required gm/ID for the input device M1 vs channel length L to meet the transistor speed target.

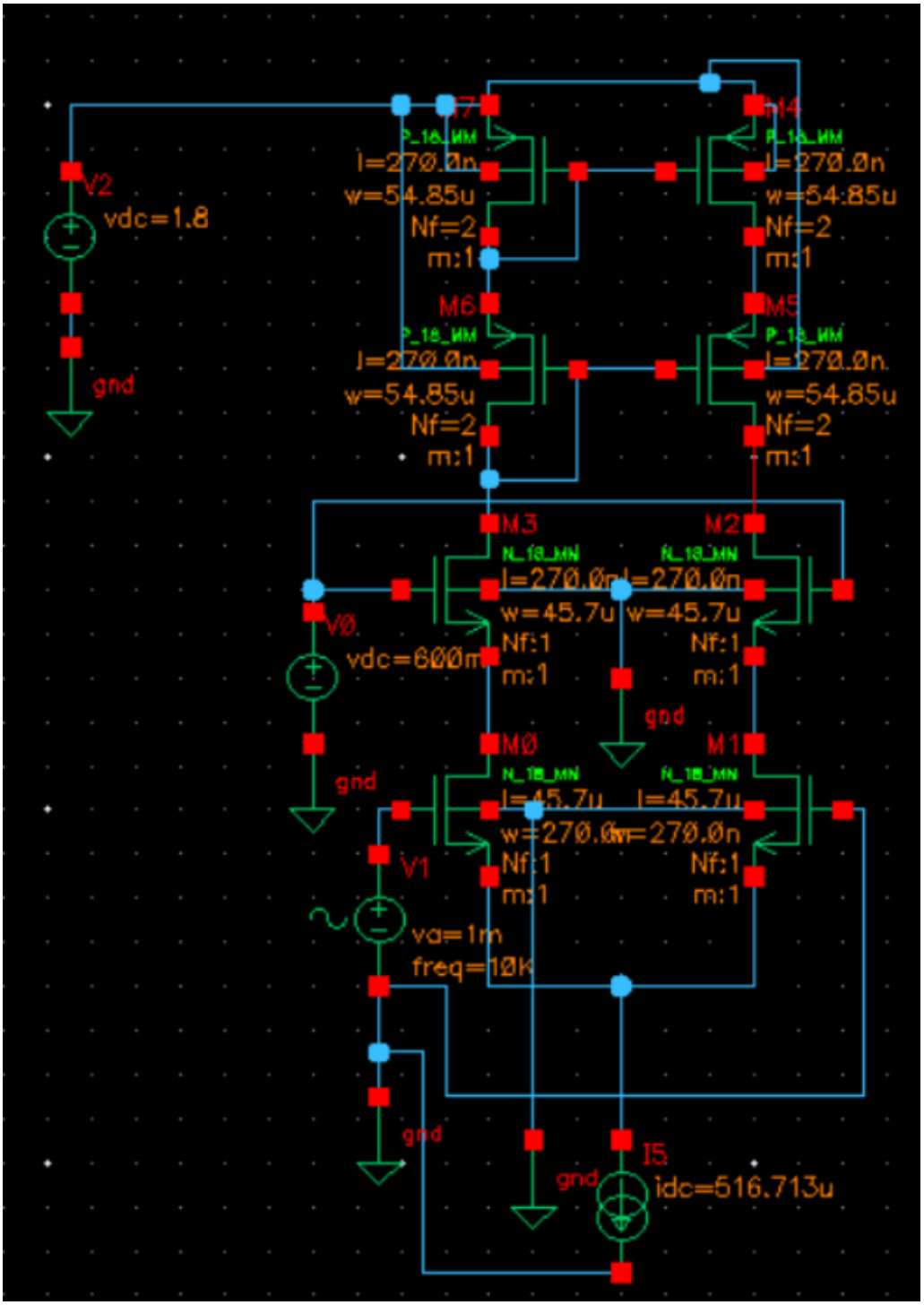

Figure 4.3: Cadence circuit schematic

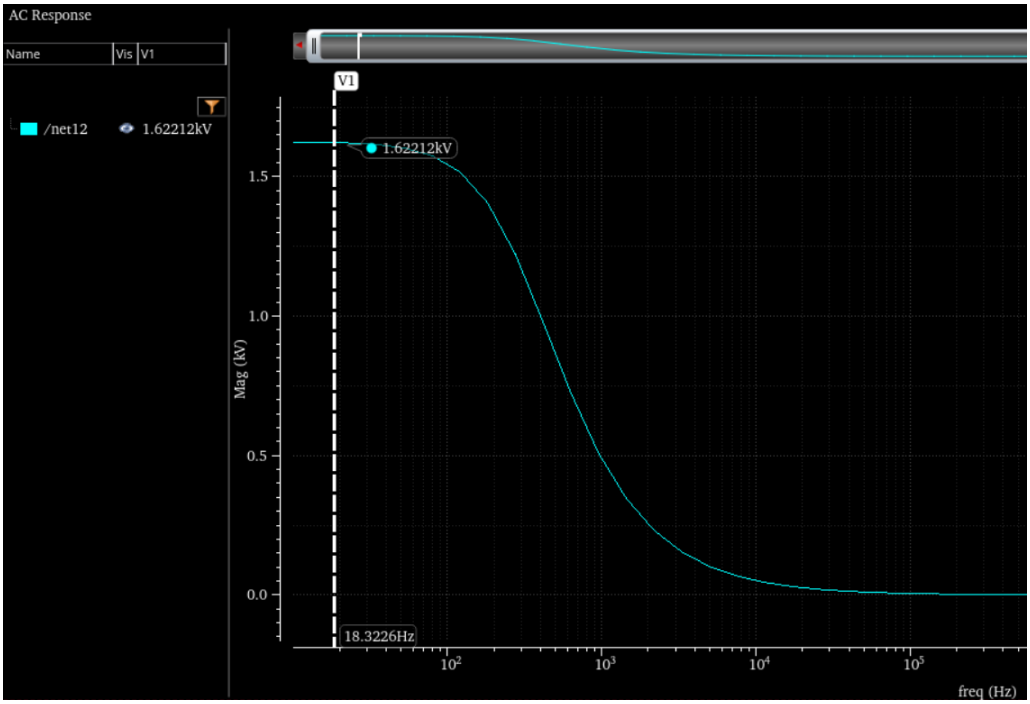

Figure 4.4: Gain vs frequency graph as plotted in Cadence Virtuoso.

4.7 Summary

This chapter demonstrates a compact gm/ID-based evaluation of gain vs length. The two generated MATLAB plots are sufficient to choose a promising length range for M1 and to understand the gm/ID operating point required to meet the transistor speed requirement implied by the target amplifier bandwidth. We achieved a maximum gain of 1.6kV at the W/L designed from our matlab code in this 8 mosfet telescopic cascode cmos circuit.