MATLAB scripts to characterize a 28nm technology

Renzo Barraza-Altamirano

2026-02-27

Introduction

The book [1] describes how to use the technique g_m/ID, together with precomputed lookup tables, to systematically design analog circuits. One key step in this process is to compute those lookup tables, and in the book associated repository [2], there are some examples about how to obtain them.

Here, I present a complement to those examples, as here the codes are adapted for a 28nm technology; they are ment to be used with MATLAB and Cadence. In contrast to the scripts in [2], these scripts characterize separatedly the PMOS and NMOS transistors and do not extract noise characteristics.

Codes for NFET

The general idea is that a MATLAB function[1] will generate the Spectre netlist, and then a separete MATLAB script will call Spectre to simulate that netlist and save the important parameters. These two codes must be in the same directory.

Function that generates the netlist for NFETs

Some caveats about this code:

- It do not capture noise parameters
- The model wrappers are hardcoded in the netlist, as I didn't find a way to include the corresponding variable to the netlist, without Spectre failling. So please, replace the line in the netlist —defined in this function— with the path to the models wrapper, and the corner you want to characterize.

```
% Function strongly based on scripts in https://github.com/bmurmann/Book-on-gm-ID-design
% 2026/02/24

function c = %% HERE PUT THE NAME OF YOUR FILE AND FUNCTION (SHOULD BE THE SAME)

% Models and file paths
c.modelinfo = 'Model info'; %% HERE PUT SOME INFO ABOUT YOUR MODELS
c.modeln = 'nfet';
c.simcmd = 'PATH_TO_SPECTRE_BIN' techsweep.scs > techsweep.out'; %%IN GENERAL, THE SPECTRE BIN IS
    UNDER A PATH LIKE: .../SPECTRE*/bin/spectre
c.outfile = 'techsweep.raw';
c.sweep = 'sweepvds_sweepvgs-sweep';
c.sweep_noise = 'sweepvds_noise_sweepvgs_noise-sweep';

% Corner dependent name for output file
```

[1] Please remember that each MATLAB function should be in a separate `.m` file, and that file must have the same name as the function itself.

```
c.corner = 'NOM';
switch c.corner
    case 'NOM'
        c.modelfile = '"PATH_TO_MODEL_WRAPPER_FILE" section=TT_CORNER'; %% THE MODEL WRAPPER SHOULD BE
            INSIDE YOUR PDK, PROBABLY IN A FOLDER CALLED MODEL AND OR SPECTRE (AS THEY ARE MODELS FOR
            SPECTRE). IN YOUR PDK DOCUMENTATION IT SHOULD BE THE DETAIL OF THE CORNER NAMES
        c.temp = 273+27;
        c.savefilen = '28nch';
    case 'SLOW'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=SS_CORNER';
        c.temp = 273+27;
        c.savefilen = '28nch_slow';
    case 'FAST'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=FF_CORNER';
        c.temp = 273;
        c.savefilen = '28nch_fast';
    case 'SLOW_FAST'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=SF_CORNER';
        c.temp = 273+125;
        c.savefilen = '28nch_slow_fast';
    case 'FAST_SLOW'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=FS_CORNER';
        c.temp = 273-40;
        c.savefilen = '28nch_fast_slow';
end

% Sweep parameters
c.VGS_step = 25e-3;
c.VDS_step = 25e-3;
c.VSB_step = 0.1;
c.VGS_max = 1.1;
c.VDS_max = 1.1;
c.VSB_max = 0.8;
c.VGS = 0:c.VGS_step:c.VGS_max;
c.VDS = 0:c.VDS_step:c.VDS_max;
c.VSB = 0:c.VSB_step:c.VSB_max;
c.LENGTH = [(0.000000030:0.000000060:0.00002)];
c.WIDTH = 0.000000500;
c.NFING = 1;

% Variable mapping
c.outvars =                 {'ID','VT','IGD','IGS','GM','GMB','GDS','CGG','CGS','CSG','CGD','CDG','
    CGB','CDD','CSS'};
c.n{1}= {'mn:ids','A',    [1    0    0    0    0    0    0    0    0    0    0    0    0    0
          0   ]};
c.n{2}= {'mn:vth','V',    [0    1    0    0    0    0    0    0    0    0    0    0    0    0
          0   ]};
c.n{3}= {'mn:igd','A',    [0    0    1    0    0    0    0    0    0    0    0    0    0    0
          0   ]};
c.n{4}= {'mn:igs','A',    [0    0    0    1    0    0    0    0    0    0    0    0    0    0
          0   ]};
c.n{5}= {'mn:gm','Ohm',   [0    0    0    0    1    0    0    0    0    0    0    0    0    0
          0   ]};
c.n{6}= {'mn:gmb','Ohm',  [0    0    0    0    0    1    0    0    0    0    0    0    0    0
          0   ]};
c.n{7}= {'mn:gds','Ohm',  [0    0    0    0    0    0    1    0    0    0    0    0    0    0
          0   ]};
c.n{8}= {'mn:cgg','F',    [0    0    0    0    0    0    0    1    0    0    0    0    0    0
          0   ]};
c.n{9}= {'mn:cgs','F',    [0    0    0    0    0    0    0    0    1    0    0    0    0    0
          0   ]};
c.n{10}={'mn:cgd','F',    [0    0    0    0    0    0    0    0    0    0    1    0    0    0
          0   ]};
c.n{11}={'mn:cgb','F',    [0    0    0    0    0    0    0    0    0    0    0    0    1    0
          0   ]};
c.n{12}={'mn:cdd','F',    [0    0    0    0    0    0    0    0    0    0    0    0    0    1
          0   ]};
c.n{13}={'mn:cdg','F',    [0    0    0    0    0    0    0    0    0    0    0    1    0    0
          0   ]};
c.n{14}={'mn:css','F',    [0    0    0    0    0    0    0    0    0    0    0    0    0    0
          1   ]};
c.n{15}={'mn:csg','F',    [0    0    0    0    0    0    0    0    0    1    0    0    0    0
          0   ]};
c.n{16}={'mn:cgsol','F',  [0    0    0    0    0    0    0    1    1    1    0    0    0    0
          1   ]};
c.n{17}={'mn:cgdol','F',  [0    0    0    0    0    0    0    1    0    0    1    1    0    1
          0   ]};
c.n{18}={'mn:cgbol','F',  [0    0    0    0    0    0    0    1    0    0    0    0    1    0
          0   ]};


c.outvars_noise = {'STH','SFL'};
c.n_noise{1}= {'mn.m1.m1:Sth', ''};
c.n_noise{2}= {'mn.m1.m1:Sfl', ''};

% Simulation netlist
netlist = sprintf([...
'//techsweep.scs \n'...
'include "PATH_TO_MODEL_WRAPPER_FILE" section=TT_CORNER \n'...
'include "techsweep_params.scs" \n'...
'save mn \n'...
'save mn \n'...
'save mn \n'...
'parameters gs=0 ds=0 \n'...
'vnoi      (vx  0)          vsource dc=0  \n'...
'vdsn      (vdn vx)         vsource dc=ds  \n'...
'vgsn      (vgn 0)          vsource dc=gs  \n'...
'vbsn      (vbn 0)          vsource dc=-sb \n'...
'mn        (vdn vgn 0 vbn) %s  l=length w=%d nf=%d \n'...
```

```
'\n'...
'simOptions options diagnose=yes gmin=1e-13 reltol=1e-4 vabstol=1e-6 iabstol=1e-10 temp=%d tnom=27
    rawfmt=psfbin rawfile="./techsweep.raw" \n'...
'sweepvds sweep param=ds start=0 stop=%d step=%d { \n'...
'   sweepvgs dc param=gs start=0 stop=%d step=%d \n'...
'}\n'...
], c.modeln, c.WIDTH, c.NFING, ...
c.temp-273, ...
c.VDS_max, c.VDS_step, ...
c.VGS_max, c.VGS_step);

% Write netlist
fid = fopen('techsweep.scs', 'w');
fprintf(fid, netlist);
fclose(fid);

return
```

Commands the multiple runs of NFET transistor

The technology characterization is done by running this script. The result, after minutes (or hours, depending on how granular are your simulations) will be a file `.mat`. The explanation about how to use that file will is in section 4.

```
% Script based on scripts in https://github.com/bmurmann/Book-on-gm-ID-design
% 2026/02/24

clearvars;
close all;

%%% Spectre matlab utilities:
spectre_matlab_path = SPECTRE_UTILITIES_DIRECTORY ; %%SPECTRE INCLUDES SOME UTILITIES THAT EASE ITS
    INTERACTION WITH MATLAB, HERE YOU MUST POINT TO THE DIRECTORY CONTAINING THOSE UTILITIES
addpath(spectre_matlab_path)

% Load configuration
c = NAME_OF_YOUR_FUNCTION_IN_PREVIOUS_SECTION;

% Write sweep info
nch.INFO   = c.modelinfo;
nch.CORNER = c.corner;
nch.TEMP   = c.temp;
nch.NFING  = c.NFING;
nch.L      = c.LENGTH';
nch.W      = c.WIDTH;
nch.VGS    = c.VGS';
nch.VDS    = c.VDS';
nch.VSB    = c.VSB';

% Simulation loop
for i = 1:length(c.LENGTH)
    str=sprintf('L = %2.3f', c.LENGTH(i));
    disp(str);
    for j = 1:length(c.VSB)
        % Write simulation parameters
        fid=fopen('techsweep_params.scs', 'w');
        fprintf(fid,'parameters length = %d\n', c.LENGTH(i));
        fprintf(fid,'parameters sb = %d\n', c.VSB(j));
        fclose(fid);

        pause(5)

        % Run simulator
        [status,result] = system(c.simcmd);
        if(status)
            disp('Simulation did not run properly. Check techsweep.out.')
            return;
        end

        % Initialize data blocks
        for m = 1:length(c.outvars)
            nch.(c.outvars{m})(i,:,:,j) = zeros(length(c.VGS), length(c.VDS));
        end

        % Read and store results
        for k = 1:length(c.n)
            params_n = c.n{k};
            struct_n = cds_srr(c.outfile, c.sweep, params_n{1}); % Function that comes with Spectre;
                you need to add Spectre's path
            values_n = struct_n.(params_n{2});
            for m = 1:length(c.outvars)
                nch.(c.outvars{m})(i,:,:,j)  = squeeze(nch.(c.outvars{m})(i,:,:,j)) + values_n*
                    params_n{3}(m);
            end
        end
    end
end

save(c.savefilen, 'nch');
```

Codes for PFET

For these codes apply the same sentences than for the NFET ones: the general idea is that a MATLAB function[2] will generate the Spectre netlist, and then a separete MATLAB script will call Spectre to simulate that netlist and save the important parameters. These two codes must be in the same directory; to keep clean directories, I recommend to place these codes in a different folder than the NFET ones.

Function that generates the netlist for PFETs

Some caveats about this code:

- It do not capture noise parameters
- The model wrappers are hardcoded in the netlist, as I didn't find a way to include the corresponding variable to the netlist, without Spectre failling. So please, replace the line in the netlist —defined in this function— with the path to the models wrapper, and the corner you want to characterize.

```
% Function strongly based on scripts in https://github.com/bmurmann/Book-on-gm-ID-design
% 2026/02/25

function c = %% HERE PUT THE NAME OF YOUR FILE AND FUNCTION (SHOULD BE THE SAME)

% Models and file paths
c.modelinfo = 'Model info'; %% HERE PUT SOME INFO ABOUT YOUR MODELS
c.modelp = 'pfet';
c.simcmd = 'PATH_TO_SPECTRE_BIN' techsweep.scs > techsweep.out'; %%IN GENERAL, THE SPECTRE BIN IS
    UNDER A PATH LIKE: .../SPECTRE*/bin/spectre
c.outfile = 'techsweep.raw';
c.sweep = 'sweepvds_sweepvgs-sweep';
c.sweep_noise = 'sweepvds_noise_sweepvgs_noise-sweep';

% Corner dependent parameters
c.corner = 'NOM';
switch c.corner
    case 'NOM'
        c.modelfile = '"PATH_TO_MODEL_WRAPPER_FILE" section=TT_CORNER'; %% THE MODEL WRAPPER SHOULD BE
            INSIDE YOUR PDK, PROBABLY IN A FOLDER CALLED MODEL AND OR SPECTRE (AS THEY ARE MODELS FOR
            SPECTRE). IN YOUR PDK DOCUMENTATION IT SHOULD BE THE DETAIL OF THE CORNER NAMES
        c.temp = 273+27;
        c.savefilep = '28pch';
    case 'SLOW'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=SS_CORNER';
        c.temp = 273+27;
        c.savefilep = '28pch_slow';
    case 'FAST'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=FF_CORNER';
        c.temp = 273;
        c.savefilep = '28pch_fast';
    case 'SLOW_FAST'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=SF_CORNER';
        c.temp = 273+125;
        c.savefilep = '28pch_slow_hot';
    case 'FAST_SLOW'
        c.modelfile = ' "PATH_TO_MODEL_WRAPPER_FILE" section=FS_CORNER';
        c.temp = 273-40;
        c.savefilep = '28pch_fast_cold';
end

% Sweep parameters
c.VGS_step = 25e-3;
c.VDS_step = 25e-3;
c.VSB_step = 0.1;
c.VGS_max = 1.1;
c.VDS_max = 1.1;
c.VSB_max = 0.8;
c.VGS = 0:c.VGS_step:c.VGS_max;
c.VDS = 0:c.VDS_step:c.VDS_max;
c.VSB = 0:c.VSB_step:c.VSB_max;
c.LENGTH = [(0.000000030:0.000000060:0.00002)];
c.WIDTH = 0.000000500;
c.NFING = 5;

% Variable mapping
```

[2]Please remember that each MATLAB function should be in a separate `.m` file, and that file must have the same name as the function itself.

```
c.outvars =              {'ID','VT','IGD','IGS','GM','GMB','GDS','CGG','CGS','CSG','CGD','CDG','CGB','
    CDD','CSS'};
c.p{1}= {'mp:ids','A',   [-1    0    0     0     0    0     0     0     0     0     0     0     0
    0     0 ]}; % Negative value to avoid record IDS negative currents
c.p{2}= {'mp:vth','V',   [0     1    0     0     0    0     0     0     0     0     0     0     0     0
          0 ]};
c.p{3}= {'mp:igd','A',   [0     0    1     0     0    0     0     0     0     0     0     0     0     0
          0 ]};
c.p{4}= {'mp:igs','A',   [0     0    0     1     0    0     0     0     0     0     0     0     0     0
          0 ]};
c.p{5}= {'mp:gm','Ohm',  [0     0    0     0     1    0     0     0     0     0     0     0     0     0
          0 ]};
c.p{6}= {'mp:gmb','Ohm', [0     0    0     0     0    1     0     0     0     0     0     0     0     0
          0 ]};
c.p{7}= {'mp:gds','Ohm', [0     0    0     0     0    0     1     0     0     0     0     0     0     0
          0 ]};
c.p{8}= {'mp:cgg','F',   [0     0    0     0     0    0     0     1     0     0     0     0     0     0
          0 ]};
c.p{9}= {'mp:cgs','F',   [0     0    0     0     0    0     0     0     1     0     0     0     0     0
          0 ]};
c.p{10}={'mp:cgd','F',   [0     0    0     0     0    0     0     0     0     0     1     0     0     0
          0 ]};
c.p{11}={'mp:cgb','F',   [0     0    0     0     0    0     0     0     0     0     0     0     1     0
          0 ]};
c.p{12}={'mp:cdd','F',   [0     0    0     0     0    0     0     0     0     0     0     0     0     1
          0 ]};
c.p{13}={'mp:cdg','F',   [0     0    0     0     0    0     0     0     0     0     0     1     0     0
          0 ]};
c.p{14}={'mp:css','F',   [0     0    0     0     0    0     0     0     0     0     0     0     0     0
          1 ]};
c.p{15}={'mp:csg','F',   [0     0    0     0     0    0     0     0     0     1     0     0     0     0
          0 ]};
c.p{16}={'mp:cgsol','F', [0     0    0     0     0    0     0     1     1     1     0     0     0     0
          1 ]};
c.p{17}={'mp:cgdol','F', [0     0    0     0     0    0     0     1     0     0     1     1     0     1
          0 ]};
c.p{18}={'mp:cgbol','F', [0     0    0     0     0    0     0     1     0     0     0     0     1     0
          0 ]};

c.outvars_noise = {'STH','SFL'};
c.p_noise{1}= {'mp.m1.m1:Sth', ''};
c.p_noise{2}= {'mp.m1.m1:Sfl', ''};

% Simulation netlist
netlist = sprintf([...
'//techsweep.scs \n'...
'include "PATH_TO_MODEL_WRAPPER_FILE" section=TT_CORNER \n'...
'include "techsweep_params.scs" \n'...
'save mp \n'...
'save mp \n'...
'save mp \n'...
'parameters gs=0 ds=0 \n'...
'vnoi      (vx  0)          vsource dc=0  \n'...
'vdsp      (vdp vx)         vsource dc=-ds \n'...
'vgsp      (vgp 0)          vsource dc=-gs \n'...
'vbsp      (vbp 0)          vsource dc=sb  \n'...
'mp        (vdp vgp 0 vbp) %s  l=length w=%d nf=%d \n'...
'\n'...
'simOptions options diagnose=yes gmin=1e-13 reltol=1e-4 vabstol=1e-6 iabstol=1e-10 temp=%d tnom=27
    rawfmt=psfbin rawfile="./techsweep.raw" \n'...
'sweepvds sweep param=ds start=0 stop=%d step=%d { \n'...
'   sweepvgs dc param=gs start=0 stop=%d step=%d \n'...
'}\n'...
], c.modelp, c.WIDTH, c.NFING, ...
c.temp-273, ...
c.VDS_max, c.VDS_step, ...
c.VGS_max, c.VGS_step);

% Write netlist
fid = fopen('techsweep.scs', 'w');
fprintf(fid, netlist);
fclose(fid);

return
```

Commands the multiple runs of PFET transistors

The technology characterization is done by running this script. The result, after minutes (or hours, depending on how granular are your simulations) will be a file `.mat`. The explanation about how to use that file will is in section 4.

```
% Function strongly based on scripts in https://github.com/bmurmann/Book-on-gm-ID-design
% 2026/02/25

clearvars;
close all;

%%% Spectre matlab utilities:
spectre_matlab_path = SPECTRE_UTILITIES_DIRECTORY ; %%SPECTRE INCLUDES SOME UTILITIES THAT EASE ITS
    INTERACTION WITH MATLAB, HERE YOU MUST POINT TO THE DIRECTORY CONTAINING THOSE UTILITIES
addpath(spectre_matlab_path)
```

```
% Load configuration
c = NAME_OF_YOUR_FUNCTION_IN_PREVIOUS_SECTION;

% Write sweep info
pch.INFO    = c.modelinfo;
pch.CORNER  = c.corner;
pch.TEMP    = c.temp;
pch.NFING   = c.NFING;
pch.L       = c.LENGTH';
pch.W       = c.WIDTH;
pch.VGS     = c.VGS';
pch.VDS     = c.VDS';
pch.VSB     = c.VSB';

% Simulation loop
for i = 1:length(c.LENGTH)
    str=sprintf('L = %2.3f', c.LENGTH(i));
    disp(str);
    for j = 1:length(c.VSB)
        % Write simulation parameters
        fid=fopen('techsweep_params.scs', 'w');
        fprintf(fid,'parameters length = %d\n', c.LENGTH(i));
        fprintf(fid,'parameters sb = %d\n', c.VSB(j));
        fclose(fid);

        pause(5)

        % Run simulator
        [status,result] = system(c.simcmd);
        if(status)
            disp('Simulation did not run properly. Check techsweep.out.')
            return;
        end

        % Initialize data blocks
        for m = 1:length(c.outvars)
            pch.(c.outvars{m})(i,:,:,j) = zeros(length(c.VGS), length(c.VDS));
        end

        % Read and store results
        for k = 1:length(c.p)
            params_p = c.p{k};
            struct_p = cds_srr(c.outfile, c.sweep, params_p{1}); % Function that comes with Spectre;
                you need to add Spectre's path
            values_p = struct_p.(params_p{2});
            for m = 1:length(c.outvars)
                pch.(c.outvars{m})(i,:,:,j)  = squeeze(pch.(c.outvars{m})(i,:,:,j)) + values_p*
                    params_p{3}(m);
            end
        end
    end
end

save(c.savefilep, 'pch');
```

Plotting gm/ID curves from .mat files

The `.mat` files, produced by the previous scripts, contain the results of all your simulations. To take those data points and generate g_m/ID curves from them, you should use a script like the following one; it is just an example, but it contains the key step of importing and calling those `.mat` files.

```
clearvars;
close all;

load 28pch.mat

gm_id = 3:0.1:20;
wt = look_up(pch, 'GM_CGG', 'GM_ID', gm_id, 'VDS', 0.6);

figure;
plot(gm_id, wt);
```

Further examples of the different plots you can generate can be found in [2].

References

[1] P. G. Jespers and B. Murmann, *Systematic design of analog CMOS circuits*. Cambridge University Press, 2017.

[2] B. Murmann, "Ancillary Material for the book "Systematic Design of Analog CMOS Circuits"." https://github.com/bmurmann/Book-on-gm-ID-design, 2025. GitHub repository.