VL502: Open-Source IC Design using IHP

Contributors:
Arnav Verma (MT2024506)
Ayushmaan Raghav (MT2024510)

Submission Date: December 6, 2024

1. IHP Details

IHP Open Source PDK

- The IHP is a non-university research establishment and a member of the Leibniz Association institutionally funded by the German federal and state governments
- Offers a 130nm BiCMOS Open Source PDK, dedicated for Analog/Digital, Mixed Signal and RF Design.
- IHP Open Source PDK project goal is to provide a fully open source Process Design Kit and related data, which can be used to create manufacturable designs at IHP's facility.

PDK Contents

- Base Cellset with Limited Set of Standard Logic Cells
  - CDL
  - GDSII
  - LEF, Tech LEF
  - Liberty
  - SPICE Netlist
  - Verilog
- IO Cellset
  - CDL
  - GDSII
  - LEF
  - Liberty
  - SPICE Netlist
  - Verilog
- SRAM Cellset
  - CDL
  - GDSII
  - LEF
  - Liberty
  - Verilog
- Primitive Devices
  - GDSII
- KLayout Tool Data

  - Layer Property and Tech Files
  - DRC Rules (Minimal/Maximal Set)
  - LVS Rules
  - PyCells (1st Priority)
  - XSection Initial Settings
- MOS/HBT/Passive Device Models for ngspice/Xyce
  - Device Models
- xschem
  - Primitive Device Symbols
  - Settings and Testbenches
- Qucs-S
  - Primitive Device Symbols
  - Settings and Testbenches
- Digital
  - Stdcells
- OpenEMS
  - Tutorials
  - Scripts
  - Documentation
- SG13G2 Process Specification and Layout Rules

2. Installation Experience

2.1 Clone the IIC-OSIC-TOOLS

```
git clone https://github.com/iic-jku/IIC-OSIC-TOOLS.git
```

IIC-OSIC-TOOLS is an all-in-one Docker container for open-source-based integrated circuit designs for analog and digital circuit flows.

2.2 Install Docker

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg
-o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc


echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Install the required packages for docker

```
sudo apt-get install docker-ce docker-ce-cli
containerd.io docker-buildx-plugin docker-compose-plugin
```

Verify Docker Installation

```
sudo docker run hello-world
```

Add User to the Docker Group

```
newgrp docker
```

2.3 Start the script using Bash

```
./start_x.sh
```

2.4 Available PDKs and Environment Variables

As of the `2022.12` tag, the following open-source process-development kits (PDKs) are pre-installed. The lists below illustrates how to switch between PDKs by setting environment variables. This can also be done per project by placing the commands into a `.designinit` file, as explained below.

– **SkyWater Technologies** `sky130A`

```
export PDK=sky130A
export PDKPATH=$PDK_ROOT/$PDK
export STD_CELL_LIBRARY=sky130_fd_sc_hd
```

– **Global Foundries** `gf180mcuC`

```
export PDK=gf180mcuC
export PDKPATH=$PDK_ROOT/$PDK
export STD_CELL_LIBRARY=gf180mcu_fd_sc_mcu7t5v0
```

– **IHP Microelectronics** `sg13g2`

```
export PDK=sg13g2
export PDKPATH=$PDK_ROOT/$PDK
export STD_CELL_LIBRARY=sg13g2_stdcell
```

2.5 Installed Tools

Below is a list of the current tools already installed and ready to use.

– **amaranth**: A Python-based HDL toolchain.
– **cace**: A Python-based circuit automatic characterization engine.
– **cocotb**: Simulation library for writing VHDL and Verilog test benches in Python.
– **covered**: Verilog code coverage tool.
– **cvc**: Circuit validity checker (ERC).
– **edalize**: Python abstraction library for EDA tools.
– **fusesoc**: Package manager and build tools for SoC.
– **gaw3-xschem**: Waveform plot tool for `xschem`.
– **gdsfactory**: Python library for GDS generation.
– **gdspy**: Python module for the creation and manipulation of GDS files.
– **gds3d**: A 3D viewer for GDS files.
– **gf180mcu**: GlobalFoundries 180nm CMOS PDK.
– **ghdl**: VHDL simulator.
– **gtkwave**: Waveform plot tool for digital simulation.
– **sg13g2**: IHP Microelectronics 130nm SiGe:C BiCMOS PDK (partial PDK, not fully supported yet; `xschem` and `ngspice` simulation works, including the PSP MOSFET model).
– **irsim**: Switch-level digital simulator.
– **iverilog**: Verilog simulator.
– **hdl21**: Analog hardware description library.
– **klayout**: Layout viewer and editor for GDS and OASIS.
– **libman**: Design library manager to manage cells and views.
– **magic**: Layout editor with DRC and PEX.

- **netgen**: Netlist comparison (LVS).
- **ngspice**: SPICE analog and mixed-signal simulator with OSDI support.
- **ngspyce**: Python bindings for `ngspice`.
- **nvc**: VHDL simulator and compiler.
- **open_pdks**: PDK setup scripts.
- **openlane2**: Rewrite of OpenLane in Python (2nd generation).
- **openram**: OpenRAM Python library.
- **openroad**: RTL-to-GDS engine used by `openlane2`.
- **osic-multitool**: Collection of useful scripts and documentation.
- **padring**: Padring generation tool.
- **pulp-tools**: PULP platform tools consisting of `bender`, `morty`, `svase`, `verible`, and `sv2v`.
- **pygmid**: Python version of the gm/Id starter kit from Boris Murmann.
- **pyopus**: Simulation runner and optimization tool for analog circuits.
- **pyrtl**: Collection of classes for Pythonic RTL design.
- **pyspice**: Interface `ngspice` and `xyce` from Python.
- **pyuvm**: Universal Verification Methodology implemented in Python (instead of SystemVerilog) using `cocotb`.
- **pyverilog**: Python toolkit for Verilog.
- **RF toolkit**: Includes `FastHenry2`, `FasterCap`, `openEMS`, and `scikit-rf`.
- **qucs-s**: Simulation environment with RF emphasis.
- **riscv-pk**: RISC-V proxy kernel and boot loader.
- **rggen**: Code generation tool for configuration and status registers.
- **schemdraw**: Python package for drawing electrical schematics.
- **slang**: SystemVerilog parsing and translation (e.g., to Verilog).
- **spike**: RISC-V ISA simulator.
- **spyci**: Analyze and plot `ngspice/xyce` output data with Python.
- **surelog**: SystemVerilog parser, elaborator, and UHDM compiler.
- **vlog2verilog**: Verilog file conversion tool.
- **volare**: Version manager (and builder) for open-source PDKs.
- **risc-v toolchain**: GNU compiler toolchain for RISC-V cores.
- **siliconcompiler**: Modular build system for hardware.
- **sky130**: SkyWater Technologies 130nm CMOS PDK.
- **verilator**: Fast Verilog simulator.
- **vlsirtools**: Interchange formats for chip design.
- **xschem**: Schematic editor.
- **xyce**: Fast parallel SPICE simulator (including `xdm` netlist conversion tool).
- **yosys**: Verilog synthesis tool (with GHDL plugin for VHDL synthesis), including `eqy` (equivalence checker), `sby` (formal verification), and `mcy` (mutation coverage).

Installation and Simulation Flowchart

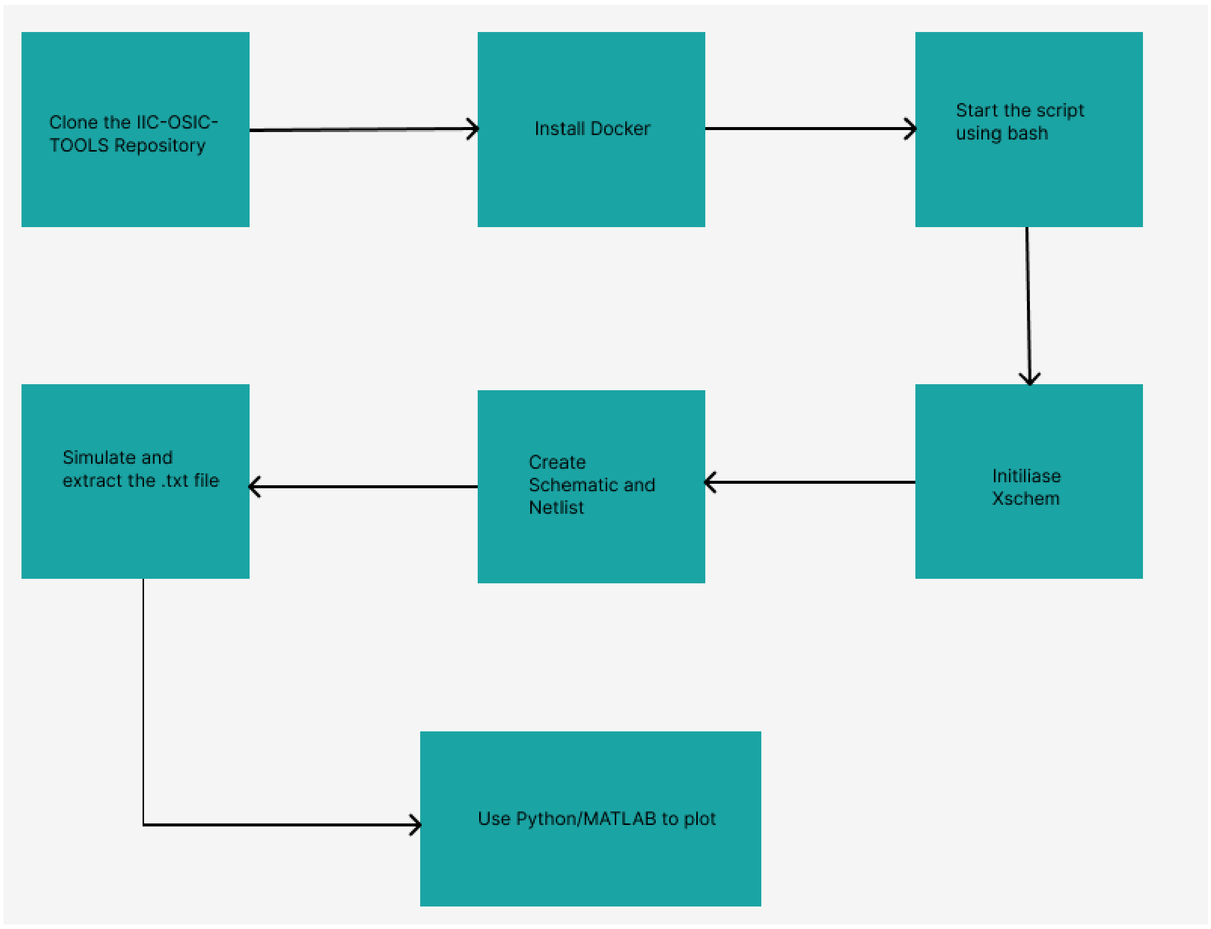
Clone the IIC-OSIC-TOOLS Repository
Install Docker
Start the script using bash
Initiliase Xschem
Create Schematic and Netlist
Simulate and extract the .txt file
Use Python/MATLAB to plot

Figure 1: Installation Flowchart

3. Technology Plots for Figures of Merit

Instructions: V_{DS} and V_{SD} for NMOS and PMOS have been swept from 0.1 V to 0.7 V in steps of 0.2 V. The lengths considered are **160 nm, 320 nm, 480 nm, 640 nm, 800 nm, and 960 nm** for a constant **Width of 1** μm**.**

NMOS Plots

1. **Vds: 0.1 V**
   (a) **gm*ro vs gm/Id**

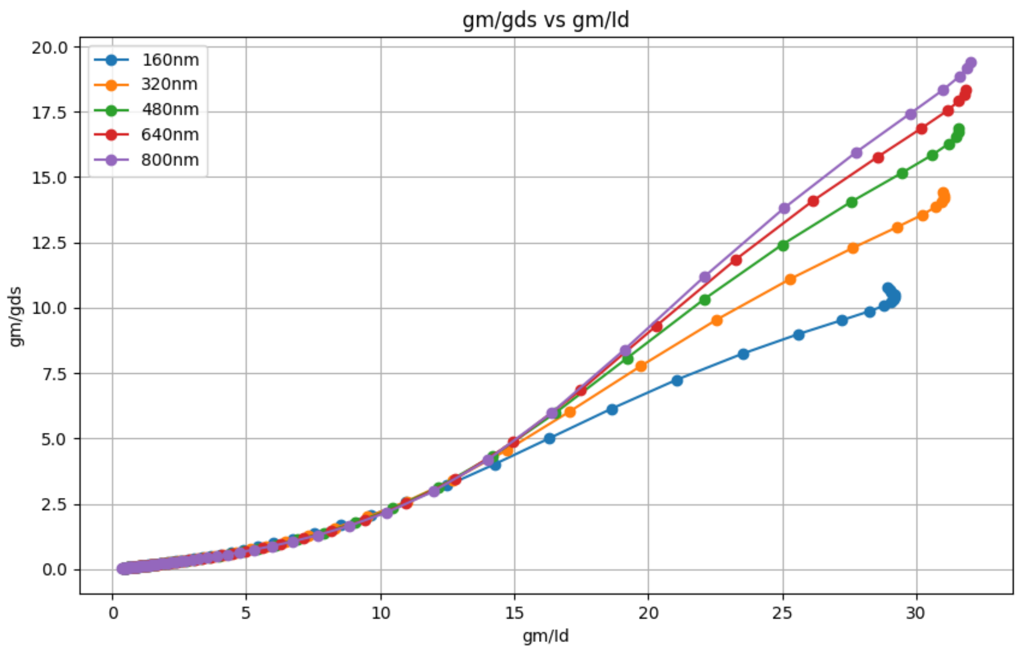

Figure 2: gm*ro vs gm/Id for NMOS at Vds = 0.1 V

(b) **ft vs gm/Id**

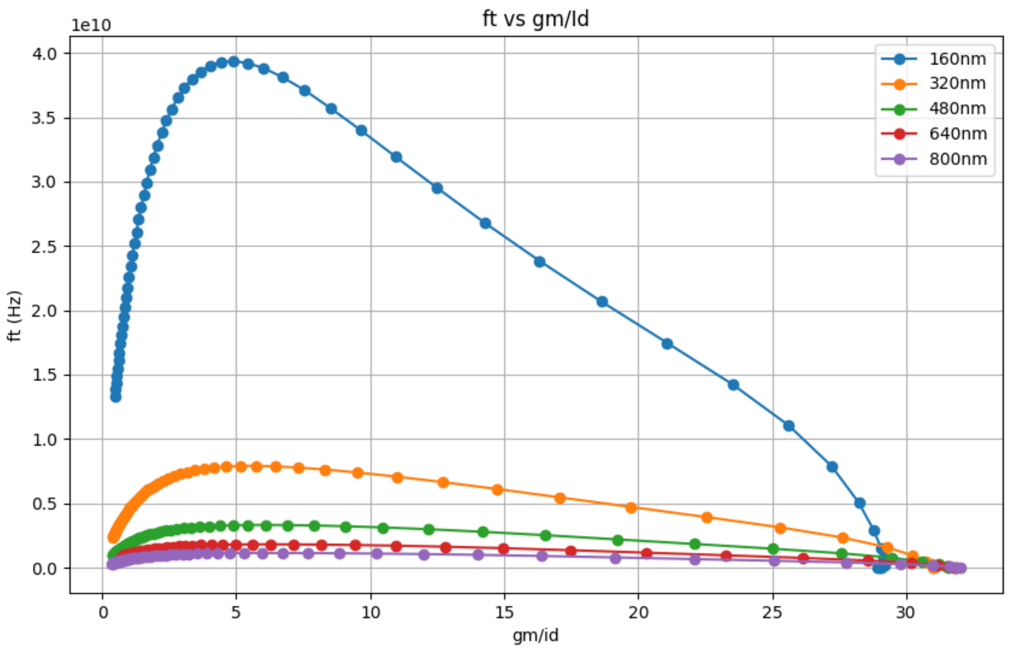

Figure 3: ft vs gm/Id for NMOS at Vds = 0.1 V

(c) **Id/W vs gm/Id**

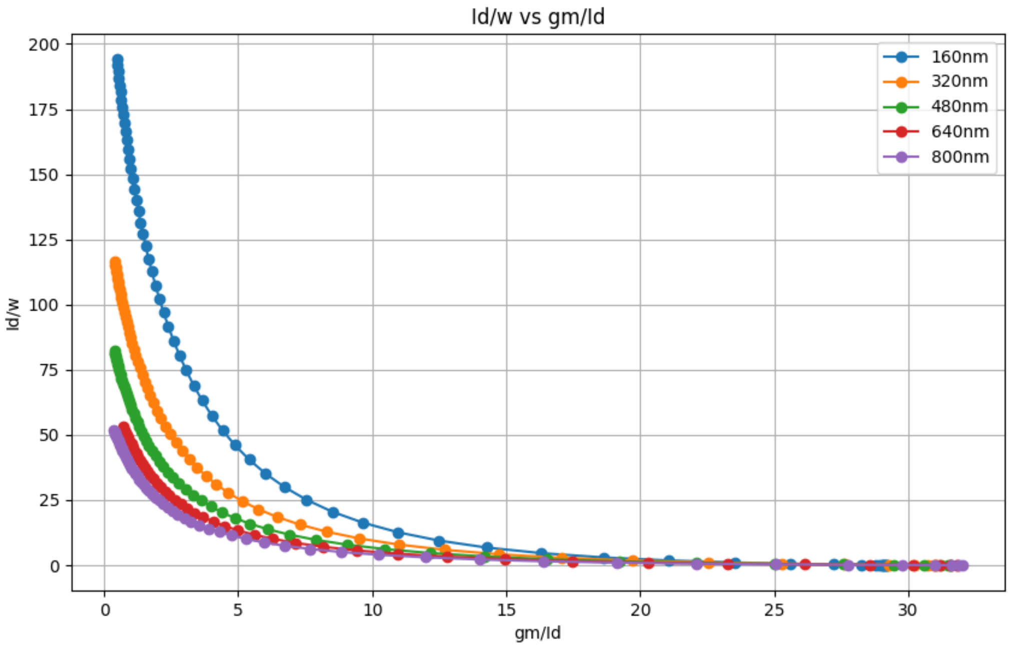

Figure 4: Id/W vs gm/Id for NMOS at Vds = 0.1 V

2. **Vds: 0.3 V**
   (a) **gm*ro vs gm/Id**

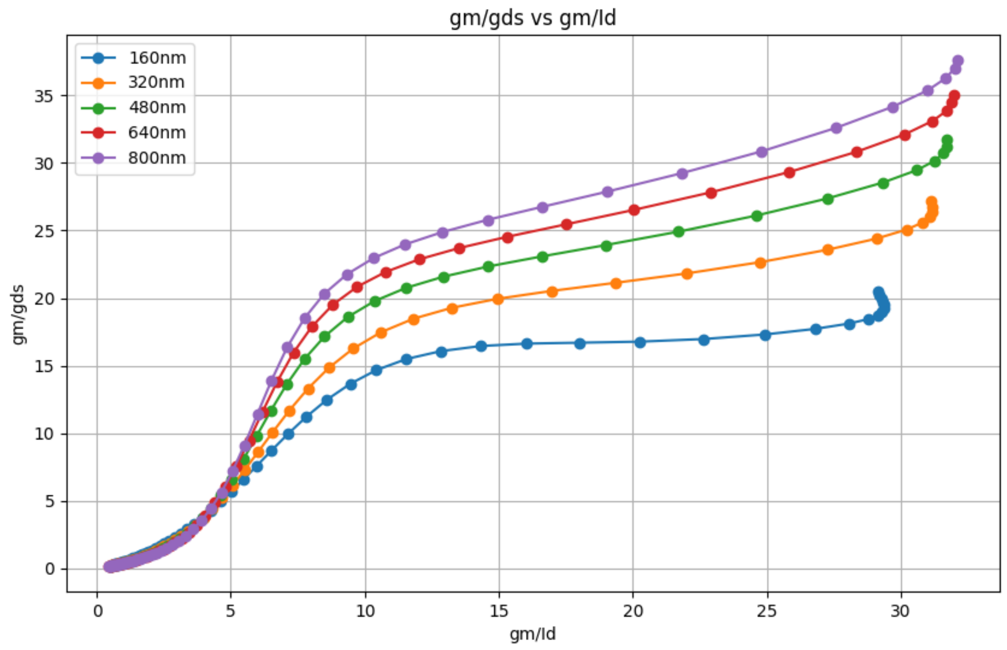

Figure 5: gm*ro vs gm/Id for NMOS at Vds = 0.3 V

   (b) **ft vs gm/Id**

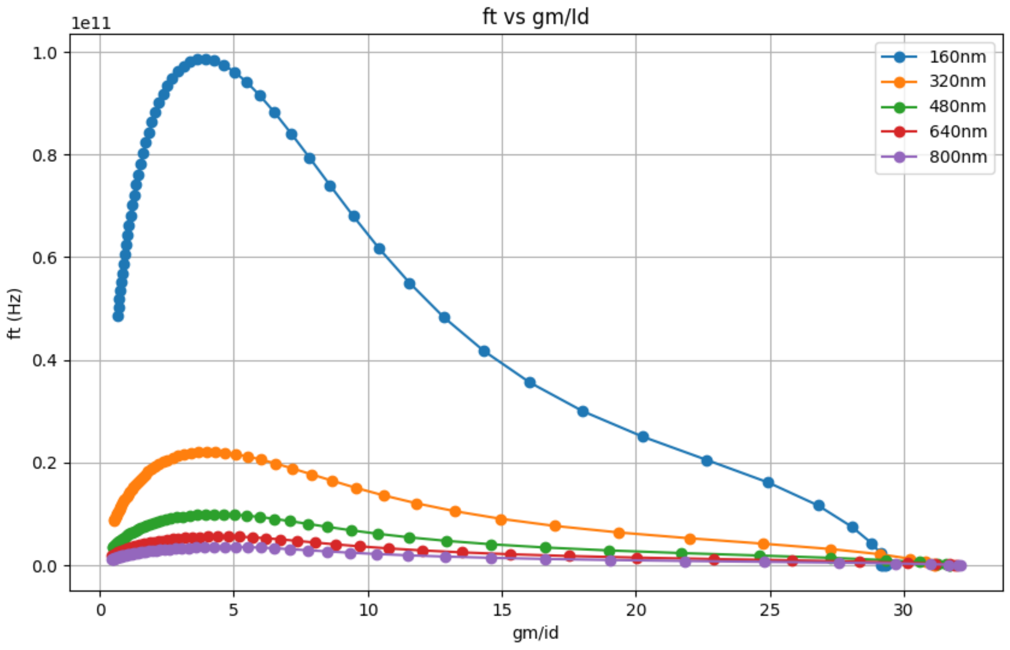

Figure 6: ft vs gm/Id for NMOS at Vds = 0.3 V

(c) **Id/W vs gm/Id**

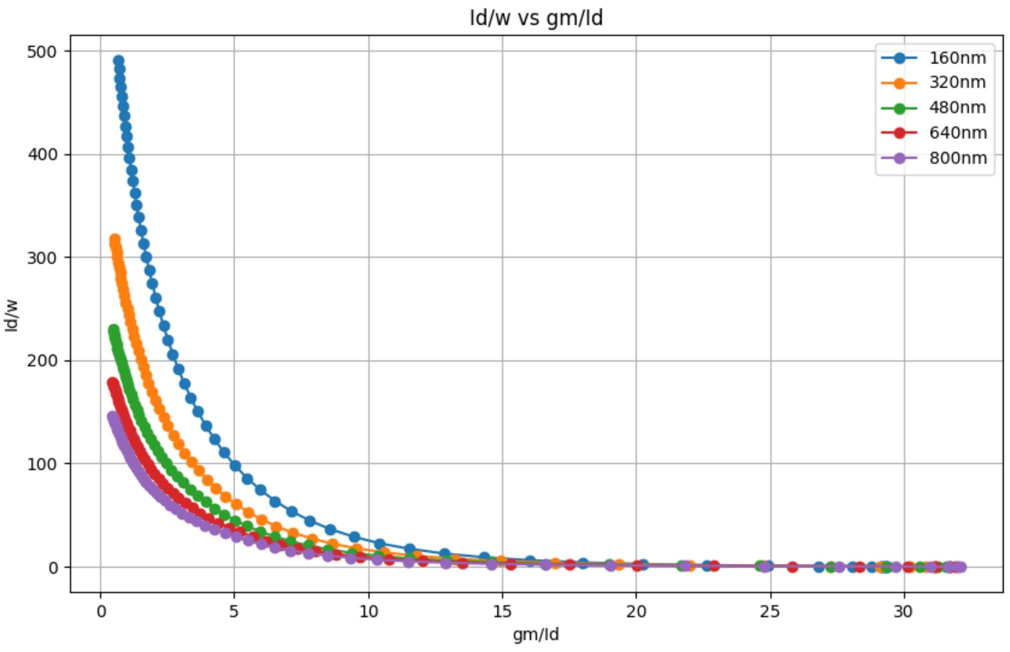

Figure 7: Id/W vs gm/Id for NMOS at Vds = 0.3 V

3. **Vds: 0.5 V**
   (a) **gm*ro vs gm/Id**

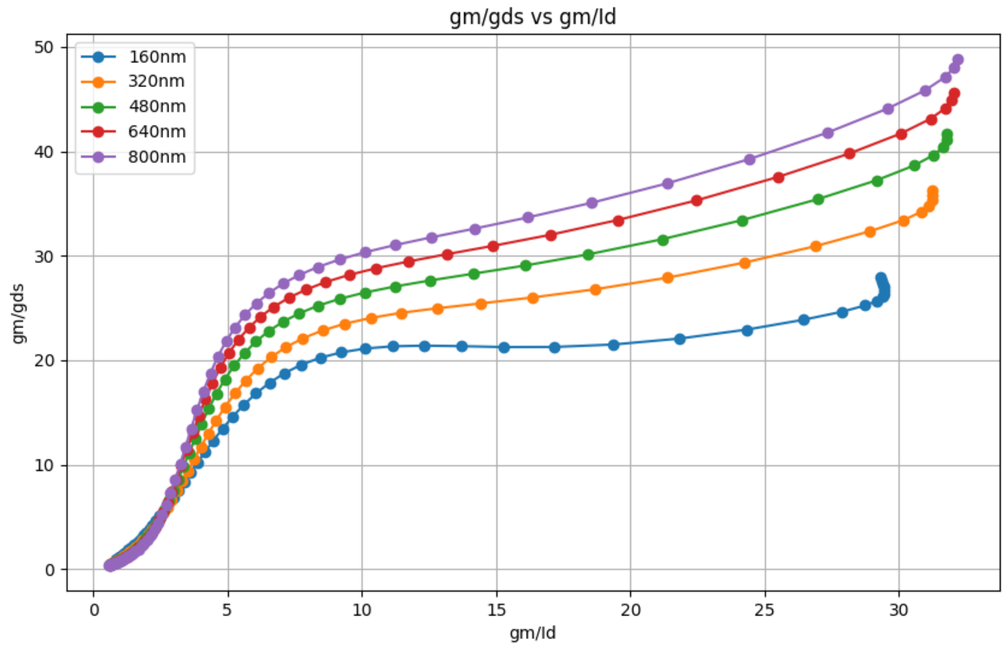

Figure 8: gm*ro vs gm/Id for NMOS at Vds = 0.5 V

(b) **ft vs gm/Id**

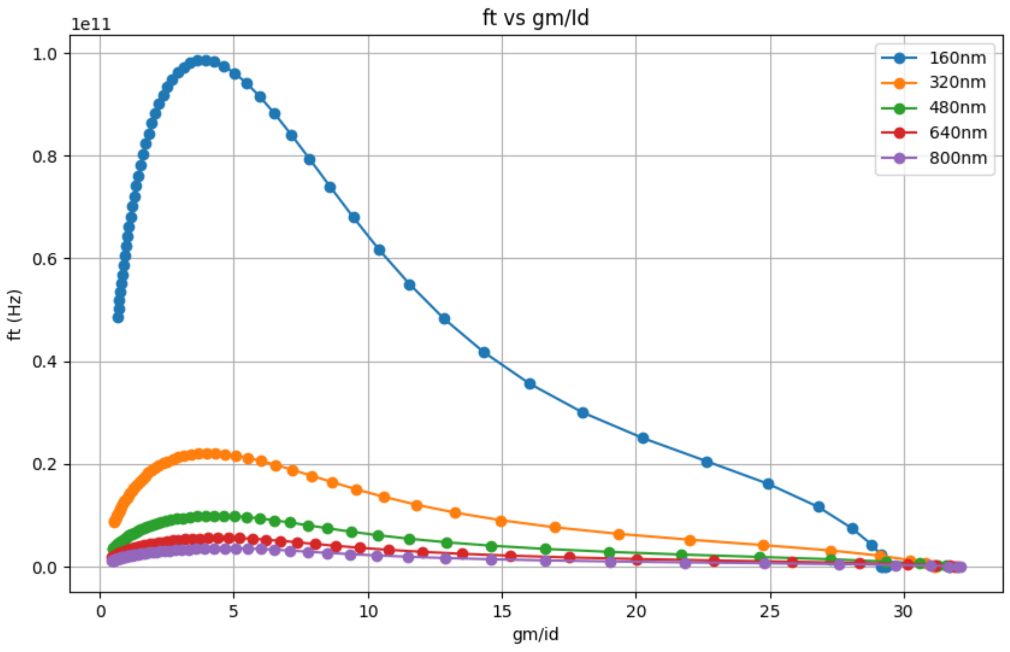

Figure 9: ft vs gm/Id for NMOS at Vds = 0.5 V

(c) **Id/W vs gm/Id**

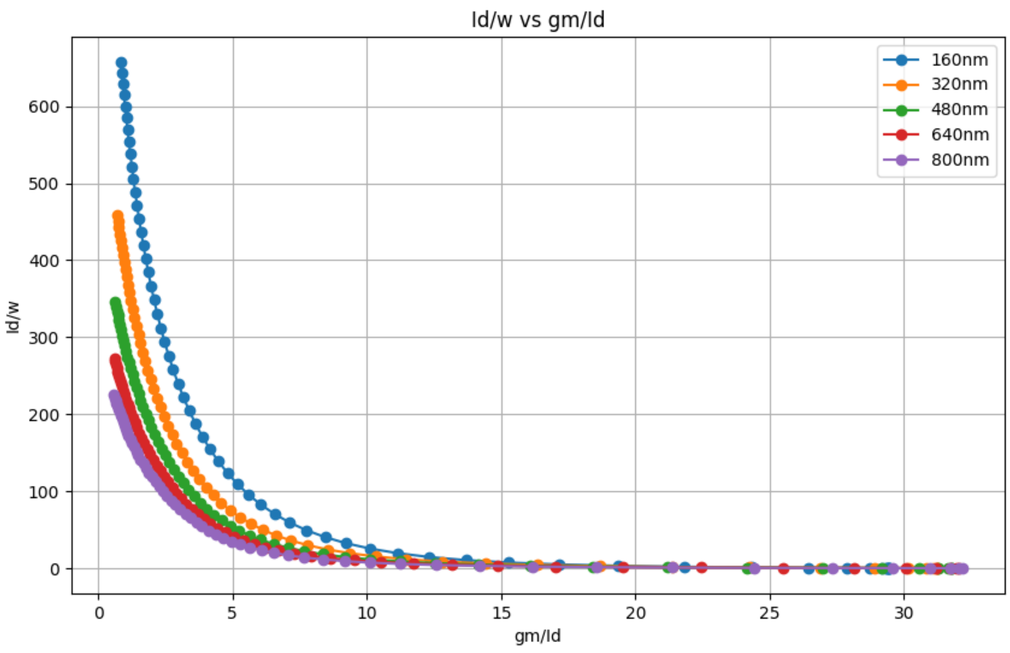

Figure 10: Id/W vs gm/Id for NMOS at Vds = 0.5 V

4. **Vds: 0.7 V**

   (a) **gm*ro vs gm/Id**

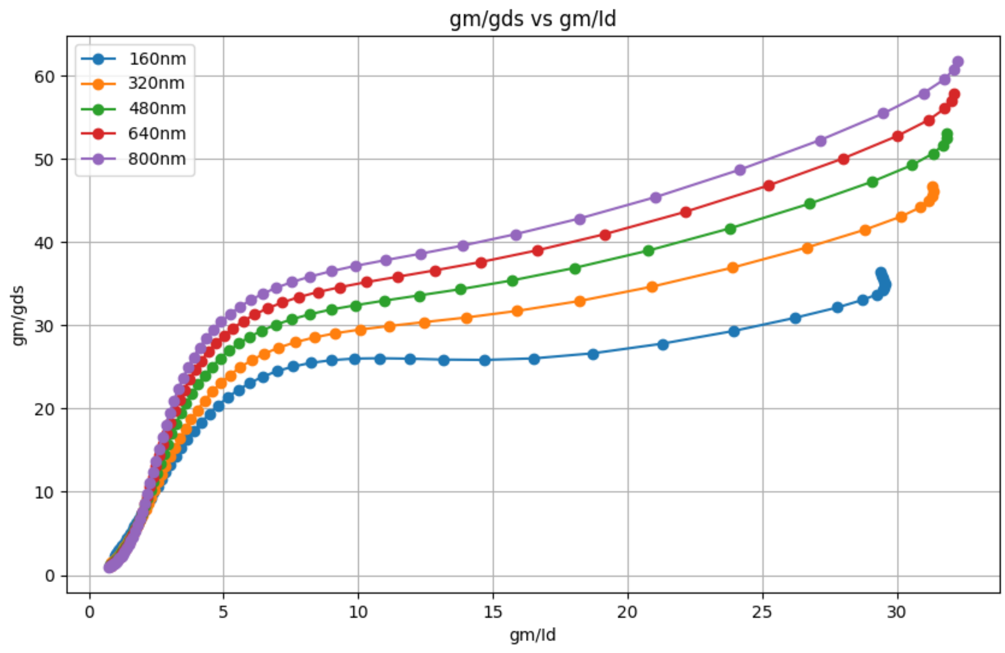

Figure 11: gm*ro vs gm/Id for NMOS at Vds = 0.7 V

   (b) **ft vs gm/Id**

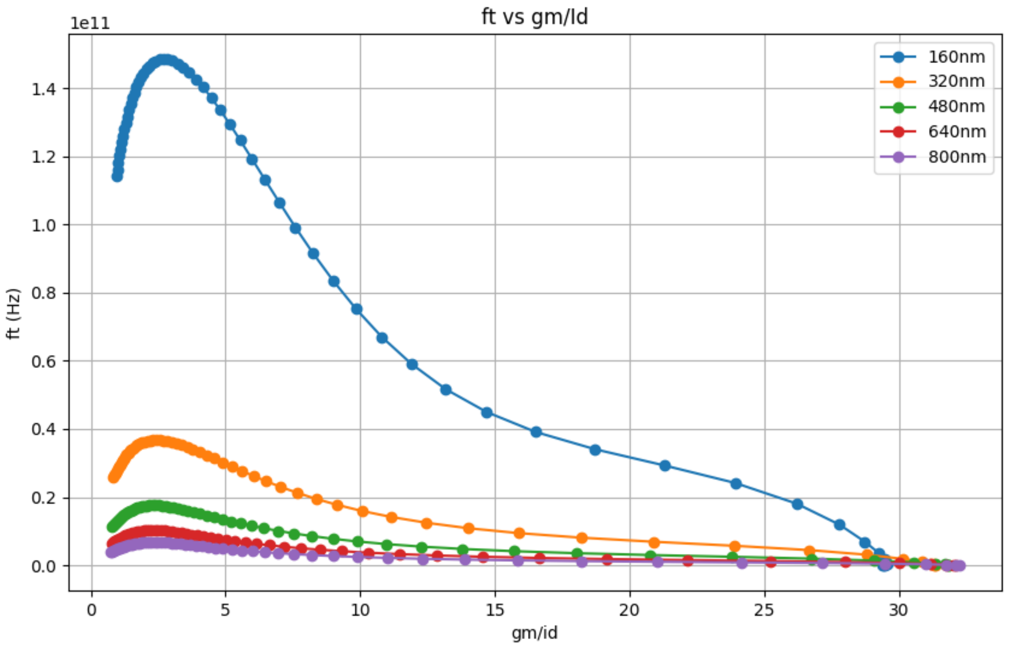

Figure 12: ft vs gm/Id for NMOS at Vds = 0.7 V

(c) **Id/W vs gm/Id**

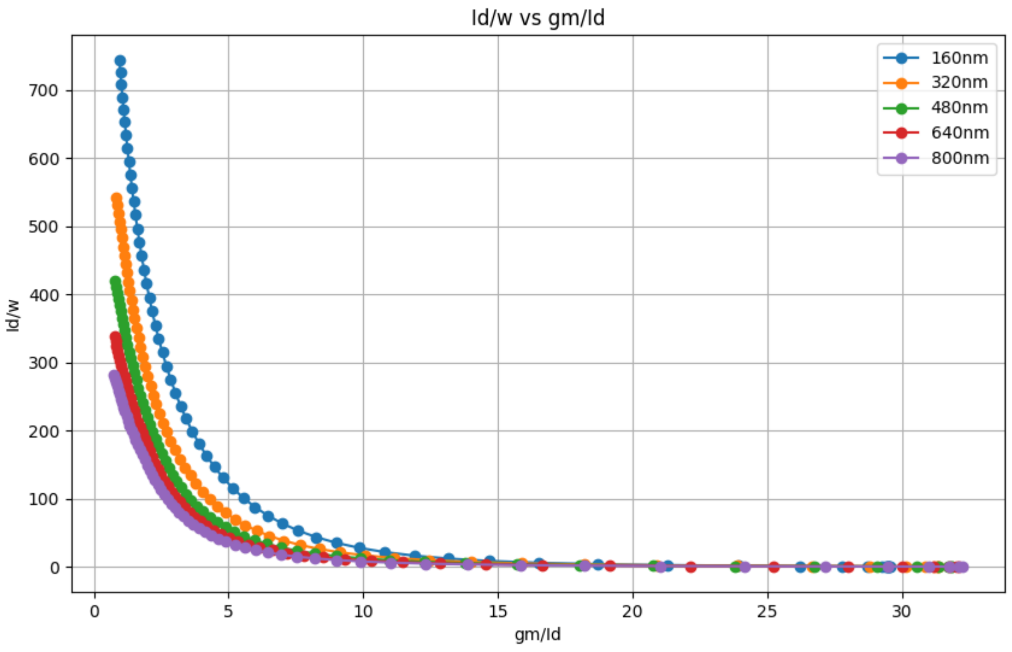

Figure 13: Id/W vs gm/Id for NMOS at Vds = 0.7 V

PMOS Plots

It has come to our findings that the PMOS Model for the FET in IHP displays obscure results. We have presented the results for PMOS for Skywater in the BONUS Section.

4. Comparison with Square Law Model

Consider IHP NMOS Model with threshold voltage of 0.35 V. Following are the parameters for the comparison:

- Length: 320nm
- $V_D = 0.3V$

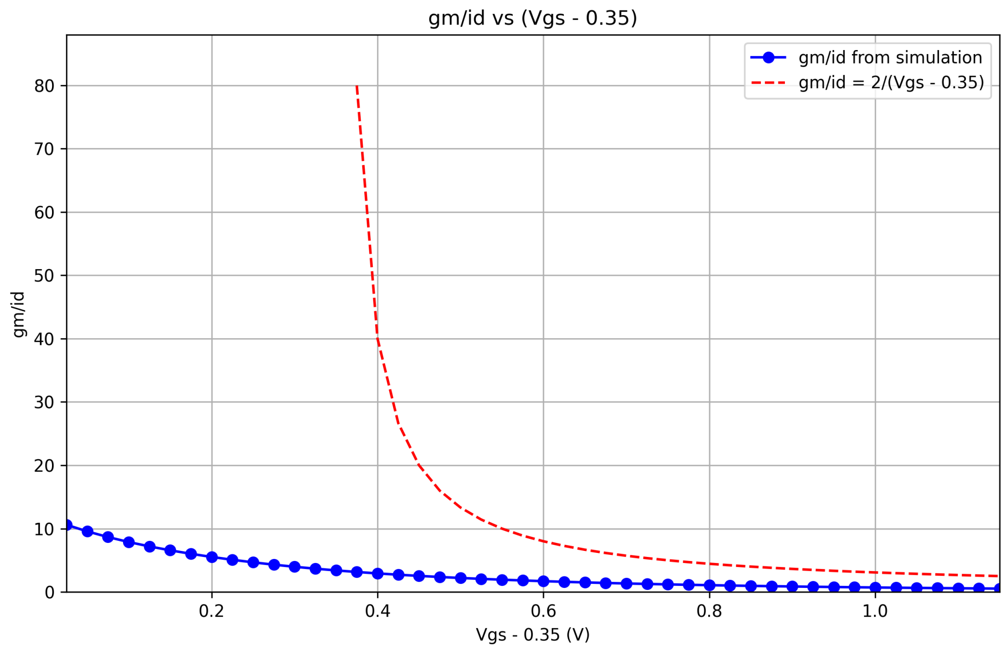

Figure 14: Comparison of g_m/I_D vs. V_{ov}: Simulation vs. Square Law.

- It is observed that the square law model breaks down in weak inversion and can be confirmed through the simulations as well.
- Why gm/Id? Why Techplots? The gm/Id methodology provides a standardized way to benchmark different devices and technologies. By comparing gm/Id values, designers can evaluate the performance of various transistors and select the most suitable one for their application. This is particularly useful for performance differentiation.

5. Optional/Bonus Credit: Skywater 130 nm Comparison

SKYWATER130 Plots for NMOS and PMOS

Schematics for NMOS and PMOS

– NMOS Schematic Example

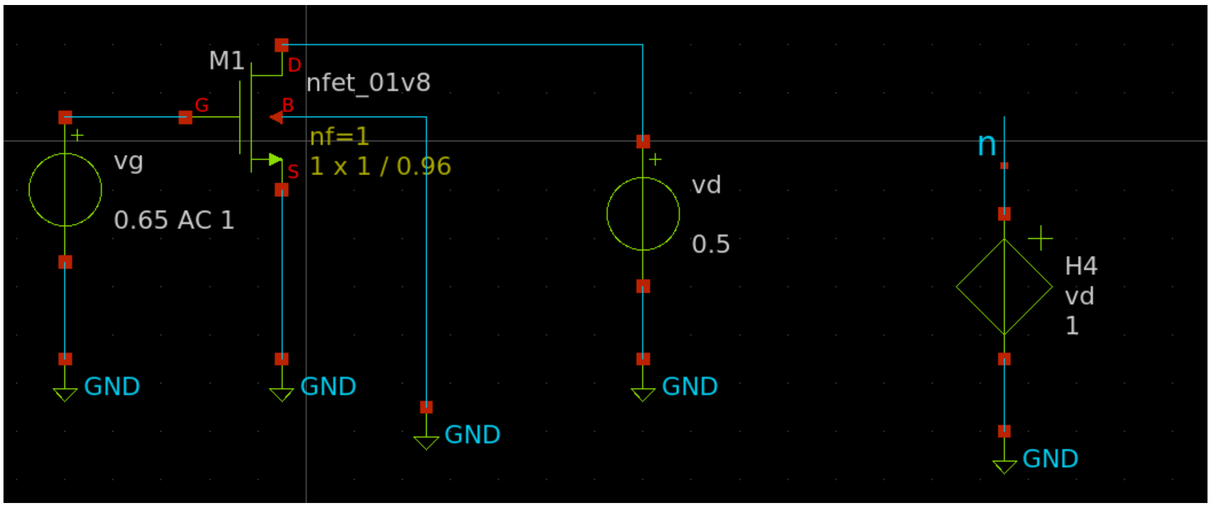

Figure 15: XSCHEM Schematic for NMOS

– PMOS Schematic Example

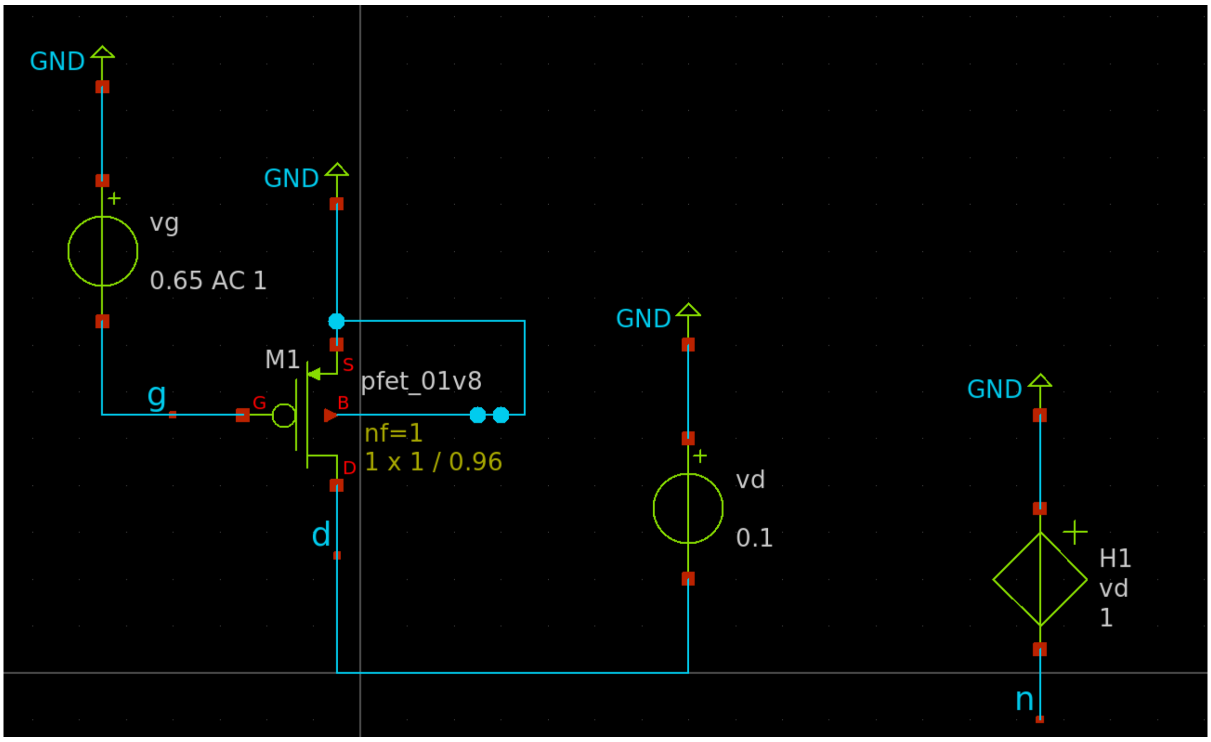

Figure 16: XSCHEM Schematic for PMOS

NGSPICE CTRL for NMOS

```
    name=NGSPICE_CTRL only_toplevel=false value="
.option sparse
.temp 27

.noise v(n) vg lin 1 1 1 1
.control
option numdgt=3
set wr_singlescale
set wr_vecnames

compose vg_vec start= 0 stop=1.5 step=25m


foreach var2 $&vg_vec
alter vg $var2


run
wrdata techsweep_sky130_fd_pr__nfet_01v8_nmos.txt noise1.all
destroy all
set appendwrite
unset set wr_vecnames

end
```

```
set appendwrite=0

reset
op
*showmod
show
write techsweep_sky130_fd_pr__nfet_01v8_nmos.raw
.endc
"
```

NGSPICE CTRL for PMOS

```
    name=NGSPICE_CTRL only_toplevel=false value="
.option sparse
.temp 27

.noise v(n) vg lin 1 1 1 1
.control
option numdgt=3
set wr_singlescale
set wr_vecnames

compose vg_vec start= 0 stop=1.5 step=25m


foreach var2 $&vg_vec
alter vg $var2


run
wrdata techsweep_sky130_fd_pr__nfet_01v8_nmos.txt noise1.all
destroy all
set appendwrite
unset set wr_vecnames

end


set appendwrite=0

reset
op
*showmod
show
write techsweep_sky130_fd_pr__nfet_01v8_nmos.raw
.endc
```

```
"
```

NOTE: The above code goes in the 'code_showm.sym' block with the name "NGSPICE_CTRL."

Saving parameters required for techplots for NMOS

```
    name=NGSPICE_SAVE only_toplevel=false value="


.save @m.xm1.msky130_fd_pr__nfet_01v8[id]
.save @m.xm1.msky130_fd_pr__nfet_01v8[cgg]
.save @m.xm1.msky130_fd_pr__nfet_01v8[gds]
.save @m.xm1.msky130_fd_pr__nfet_01v8[gm]
.save @m.xm1.msky130_fd_pr__nfet_01v8[l]
.save @m.xm1.msky130_fd_pr__nfet_01v8[vgs]
.save @m.xm1.msky130_fd_pr__nfet_01v8[vds]
.save @m.xm1.msky130_fd_pr__nfet_01v8[w]

"
```

Saving Paramters for techplots for PMOS

```
    name=NGSPICE_SAVE only_toplevel=false value="


.save @m.xm1.msky130_fd_pr__pfet_01v8[id]
.save @m.xm1.msky130_fd_pr__pfet_01v8[cgg]
.save @m.xm1.msky130_fd_pr__pfet_01v8[gds]
.save @m.xm1.msky130_fd_pr__pfet_01v8[gm]
.save @m.xm1.msky130_fd_pr__pfet_01v8[l]
.save @m.xm1.msky130_fd_pr__pfet_01v8[vgs]
.save @m.xm1.msky130_fd_pr__pfet_01v8[vds]
.save @m.xm1.msky130_fd_pr__pfet_01v8[w]

"
```

NOTE: The above code goes in the 'code_showm.sym' block with the name "NGSPICE_SAVE."

NMOS Plots

1. Vds: 0.1 V

- **gm*ro vs gm/Id**

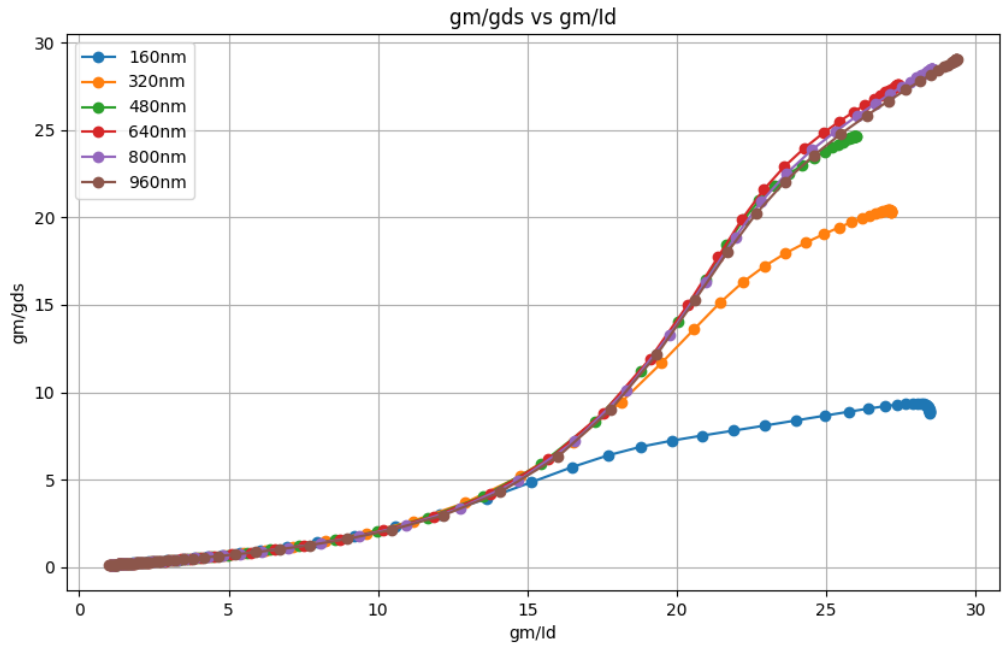

Figure 17: gm*ro vs gm/Id for NMOS at Vds = 0.1 V

– **ft vs gm/Id**

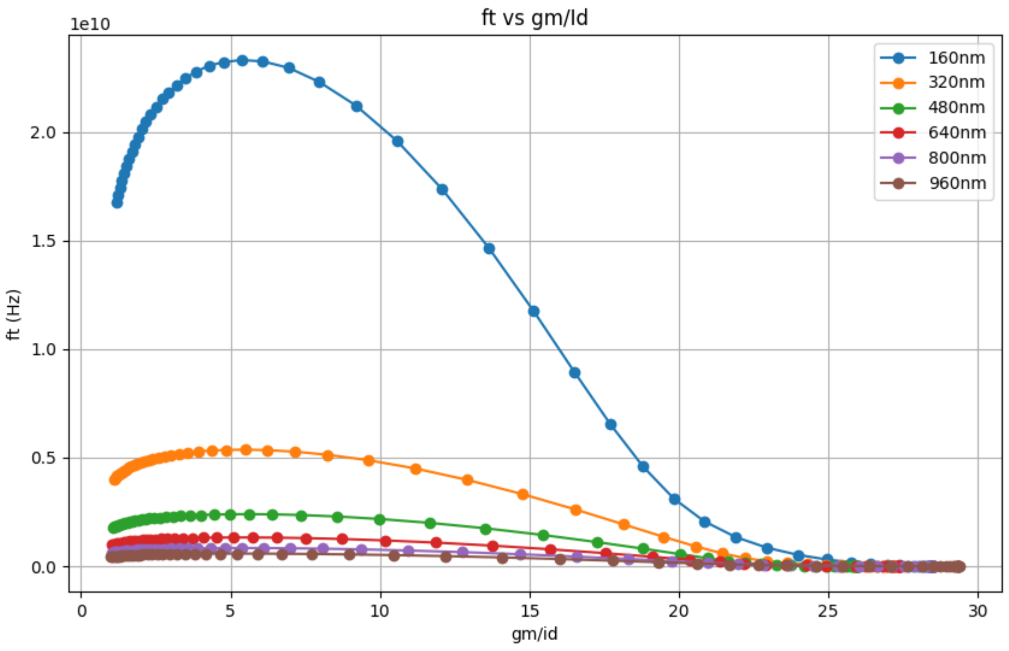

Figure 18: ft vs gm/Id for NMOS at Vds = 0.1 V

– **Id/W vs gm/Id**

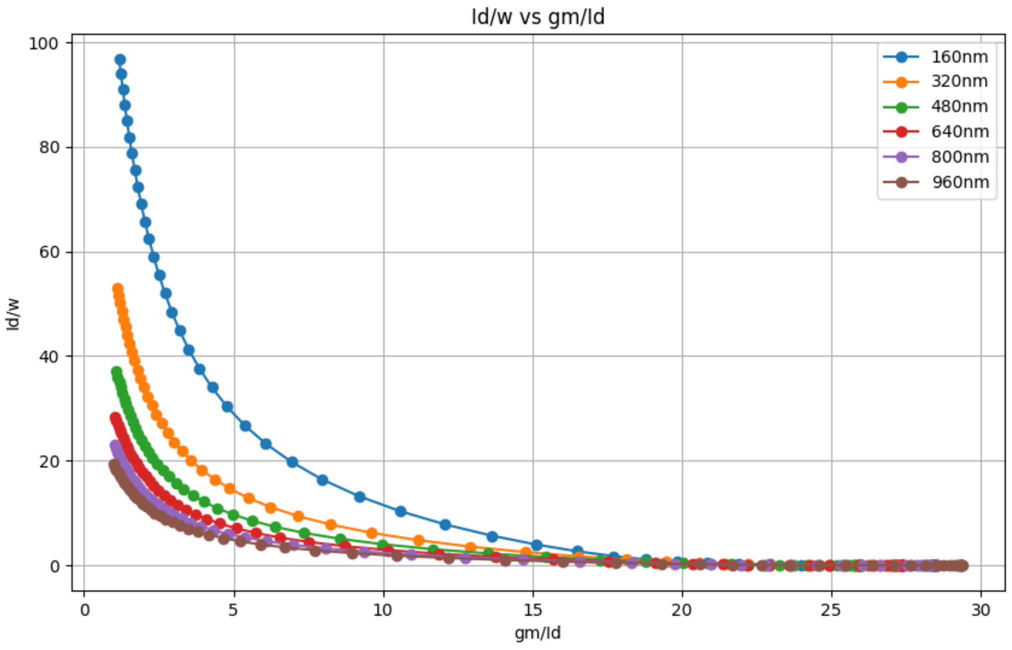

Figure 19: Id/W vs gm/Id for NMOS at Vds = 0.1 V

2. Vds: 0.3 V

– **gm*ro vs gm/Id**

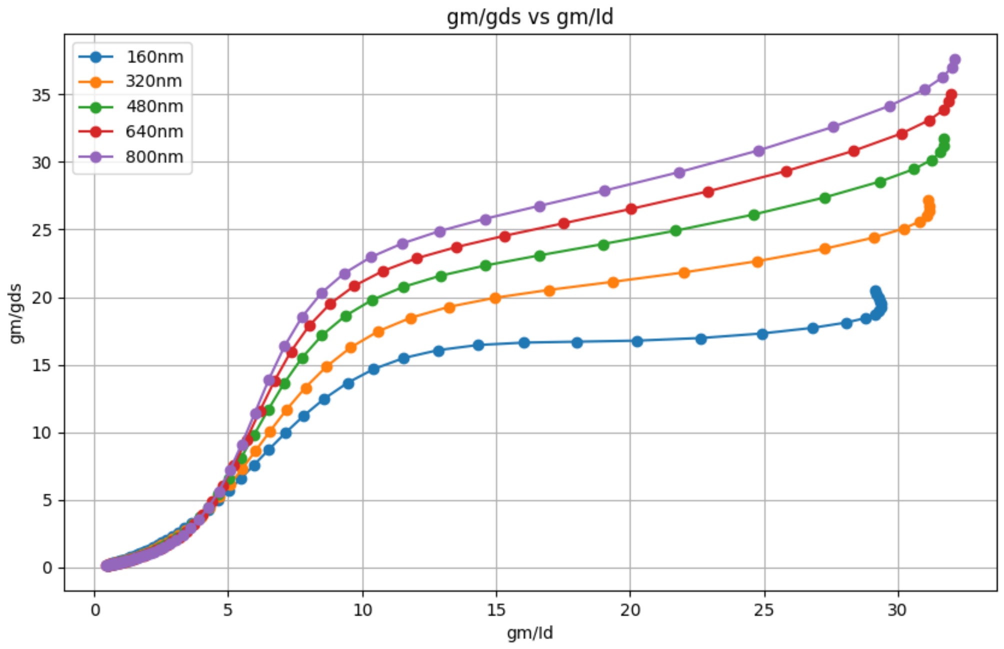

Figure 20: gm*ro vs gm/Id for NMOS at Vds = 0.3 V

– **ft vs gm/Id**

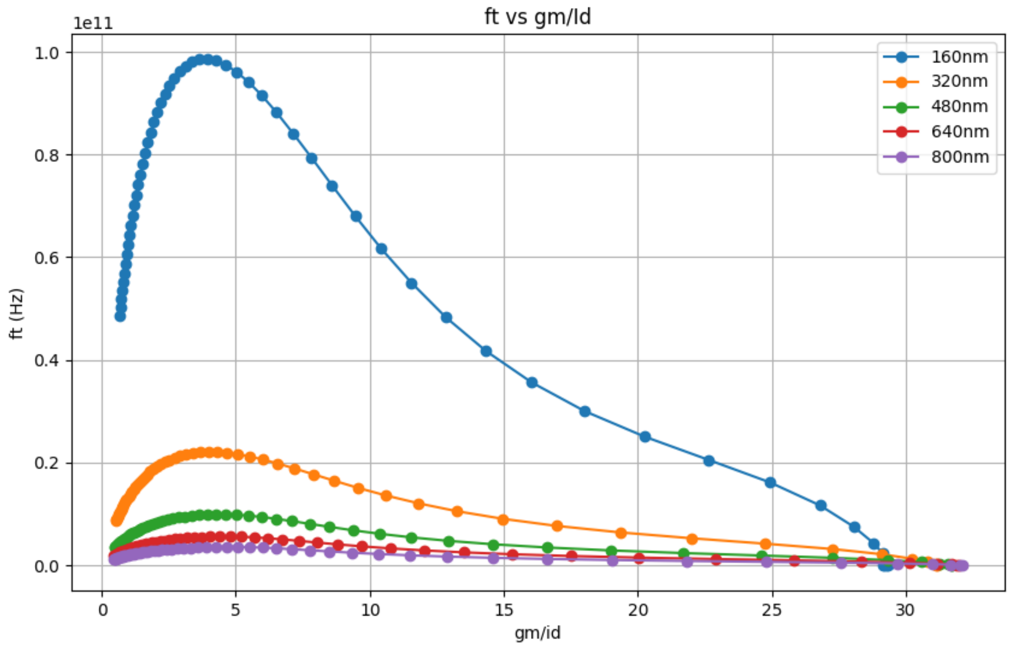

Figure 21: ft vs gm/Id for NMOS at Vds = 0.3 V

– **Id/W vs gm/Id**

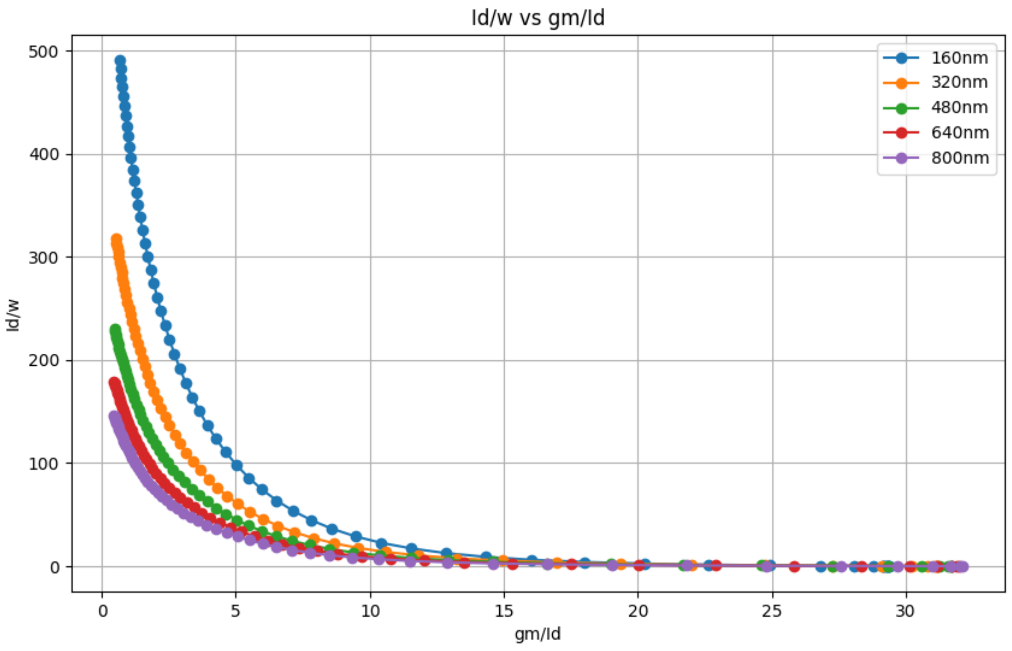

Figure 22: Id/W vs gm/Id for NMOS at Vds = 0.3 V

3. Vds: 0.5 V

– **gm*ro vs gm/Id**

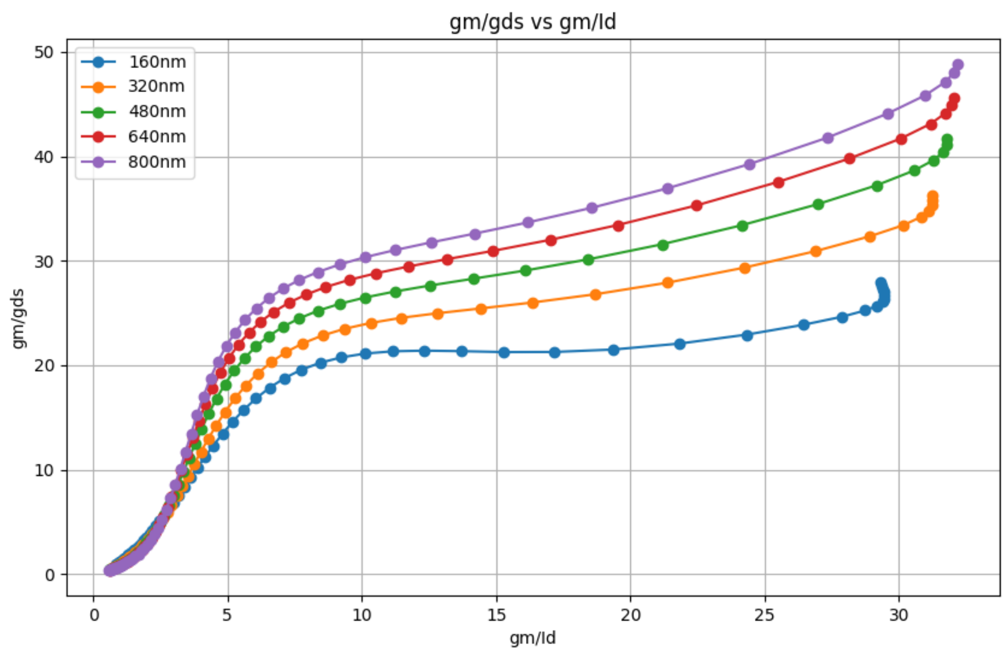

Figure 23: gm*ro vs gm/Id for NMOS at Vds = 0.5 V

– **ft vs gm/Id**

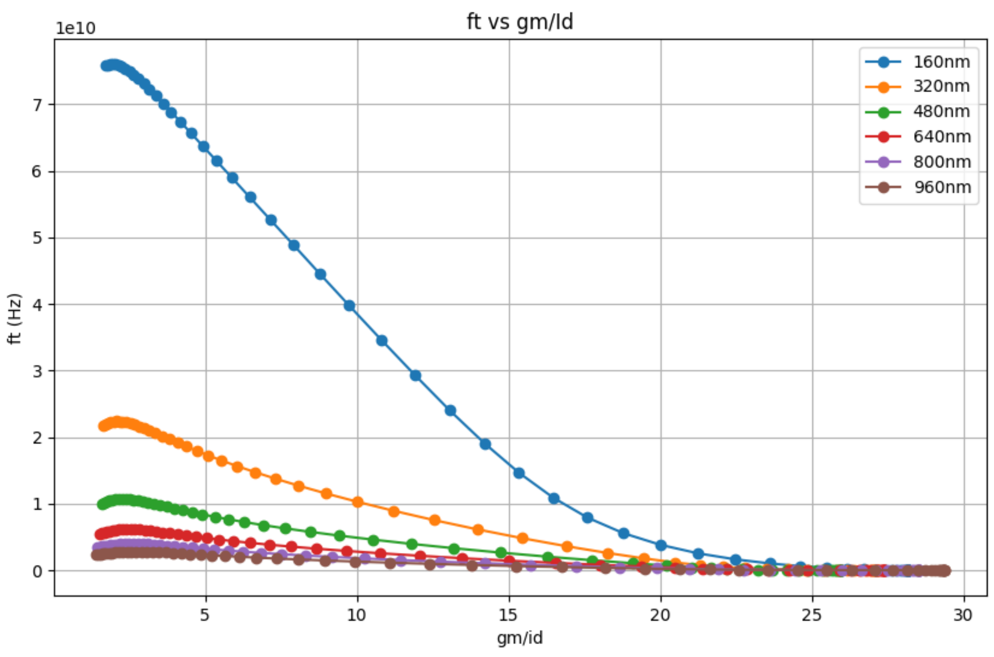

Figure 24: ft vs gm/Id for NMOS at Vds = 0.5 V

– **Id/W vs gm/Id**

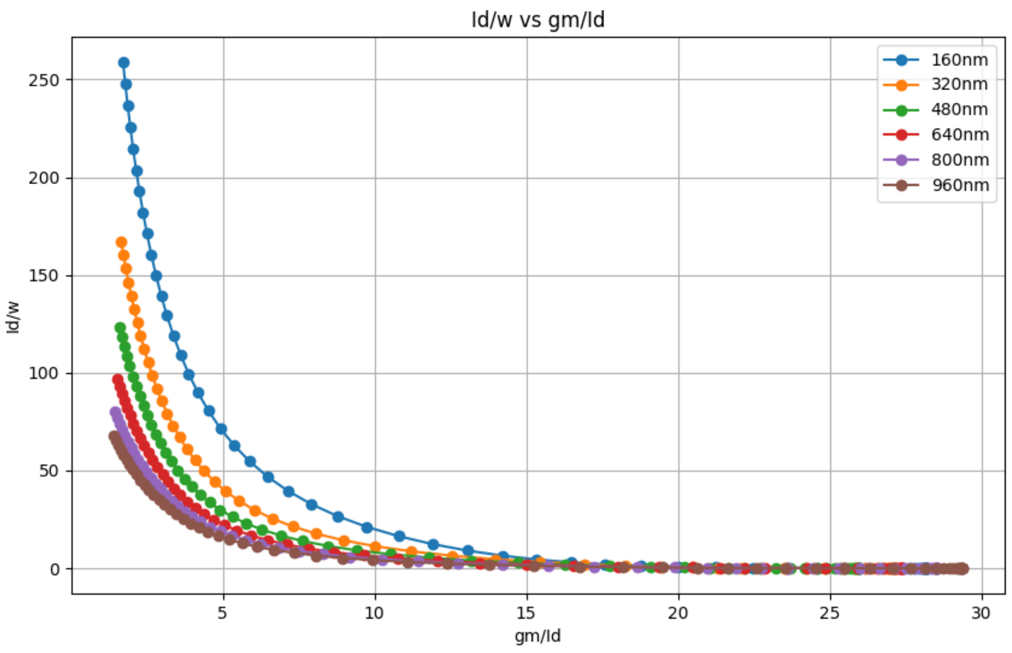

Figure 25: Id/W vs gm/Id for NMOS at Vds = 0.5 V

4. Vds: 0.7 V

– gm*ro vs gm/Id

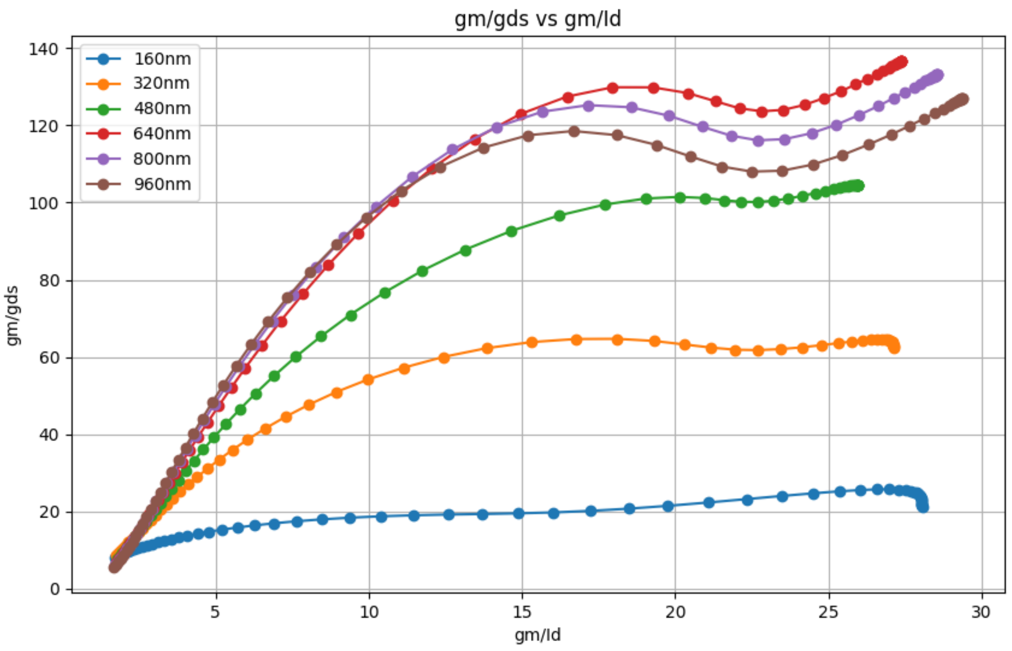

Figure 26: gm*ro vs gm/Id for NMOS at Vds = 0.7 V

– ft vs gm/Id

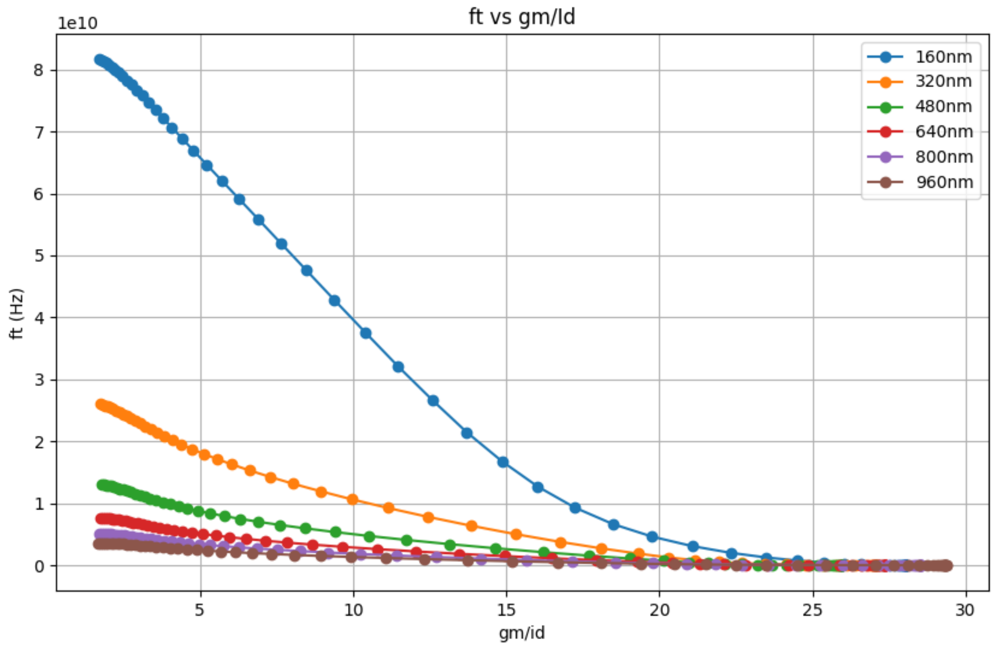

Figure 27: ft vs gm/Id for NMOS at Vds = 0.7 V

– **Id/W vs gm/Id**

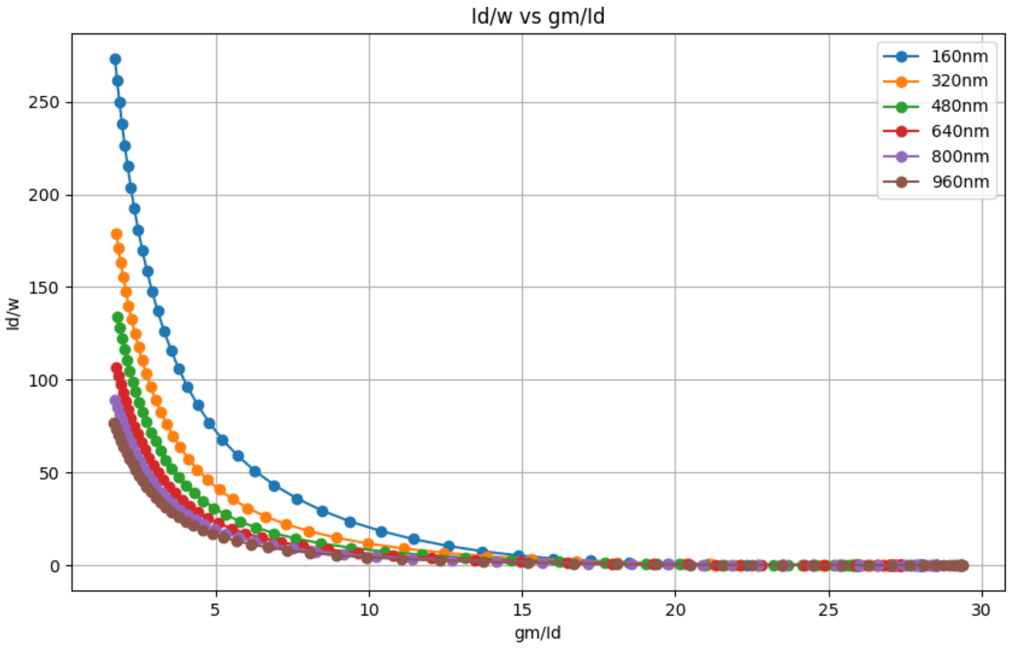

Figure 28: Id/W vs gm/Id for NMOS at Vds = 0.7 V

PMOS Plots

1. **Vsd: 0.1 V**

– **gm*ro vs gm/Id**

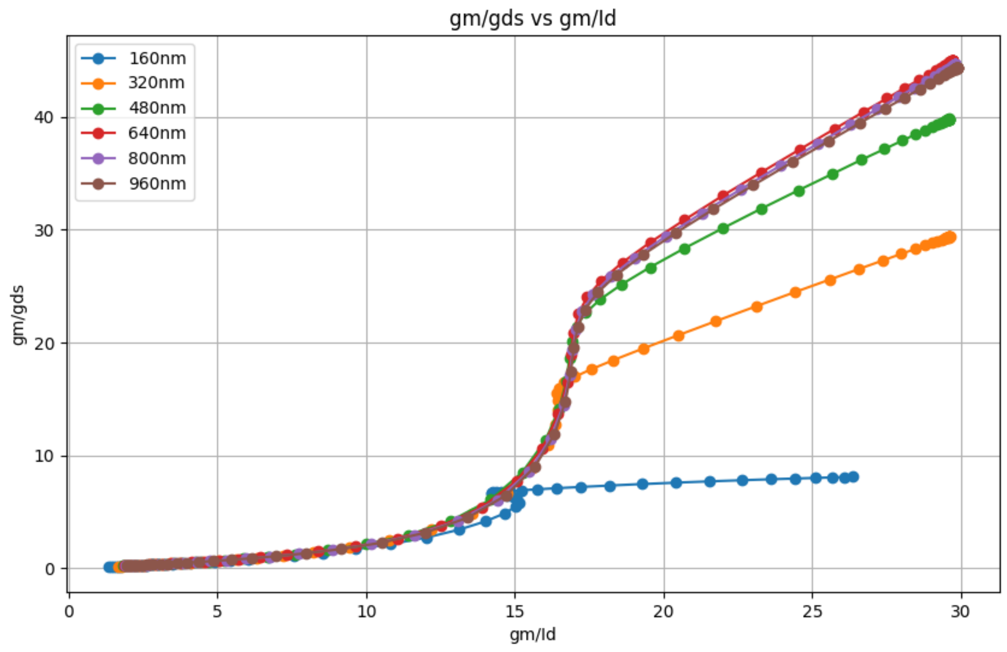

Figure 29: gm*ro vs gm/Id for PMOS at Vsd = 0.1 V

– **ft vs gm/Id**

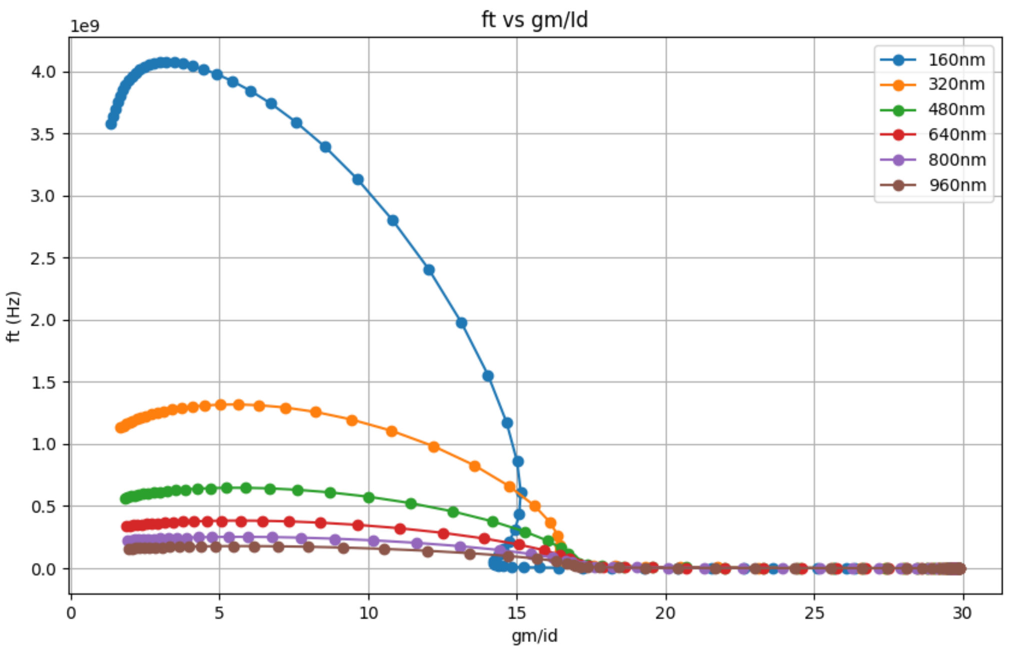

Figure 30: ft vs gm/Id for PMOS at Vsd = 0.1 V

– **Id/W vs gm/Id**

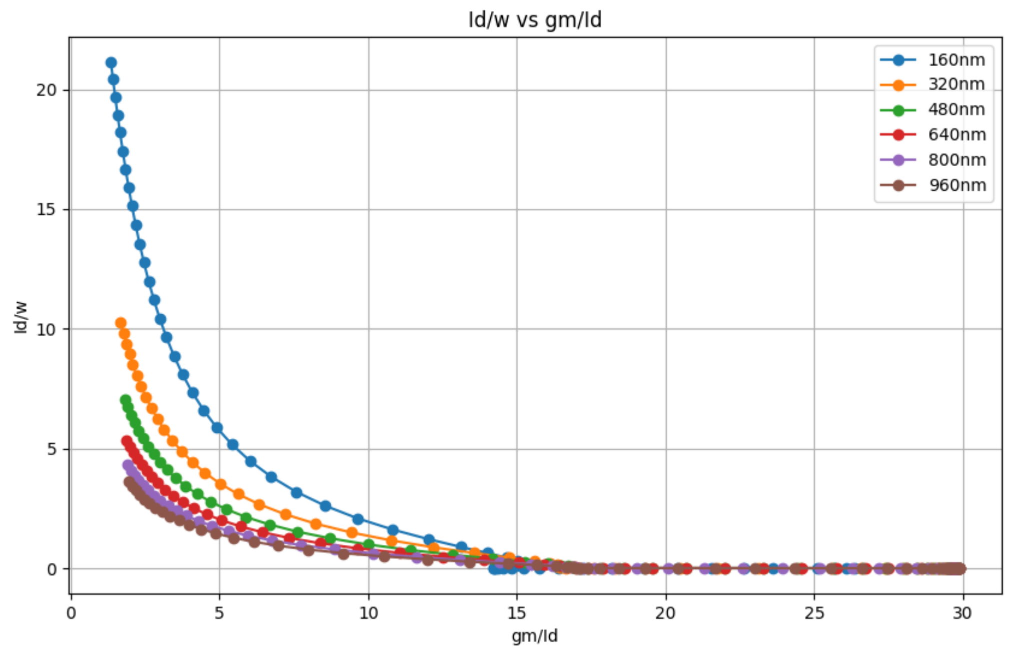

Figure 31: Id/W vs gm/Id for PMOS at Vsd = 0.1 V

2. **Vsd: 0.3 V**
   - **gm*ro vs gm/Id**

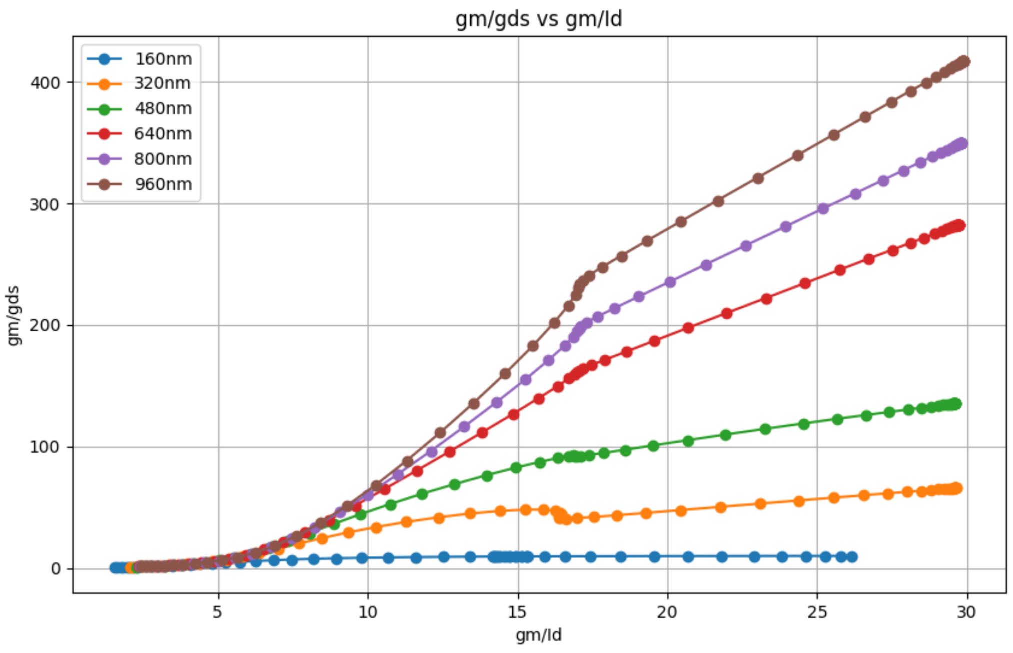

Figure 32: gm*ro vs gm/Id for PMOS at Vsd = 0.3 V

   - **ft vs gm/Id**

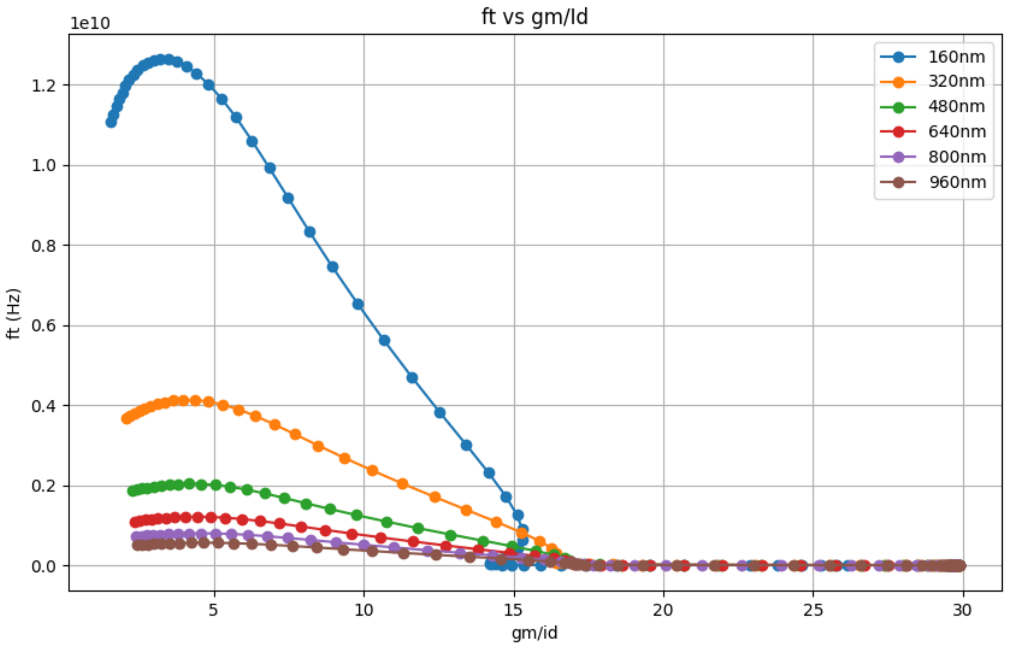

Figure 33: ft vs gm/Id for PMOS at Vsd = 0.3 V

- **Id/W vs gm/Id**

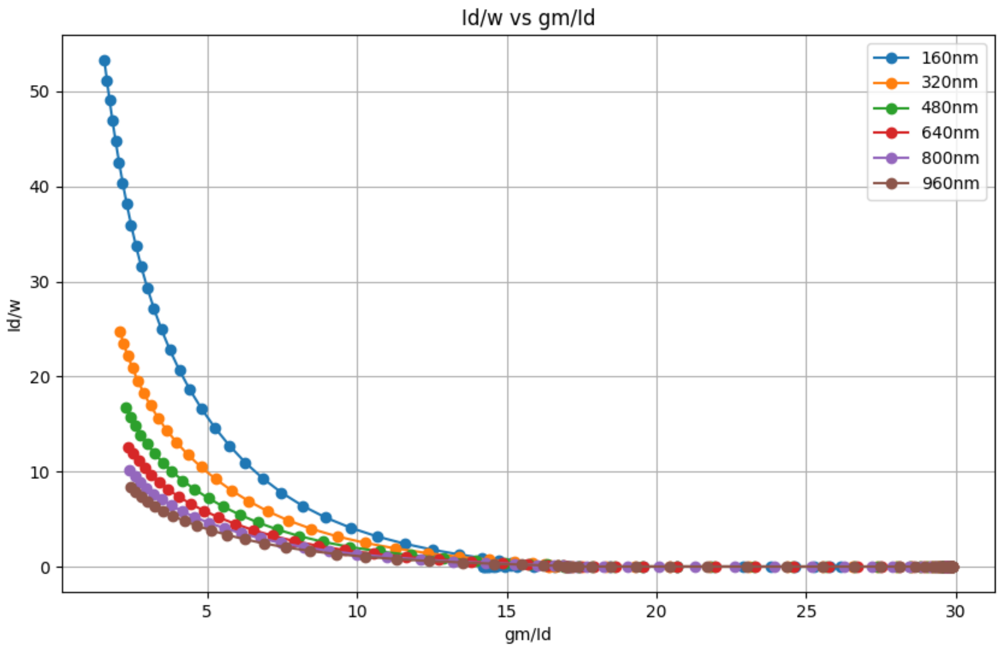

Figure 34: Id/W vs gm/Id for PMOS at Vsd = 0.3 V

3. **Vsd: 0.5 V**
   - **gm*ro vs gm/Id**

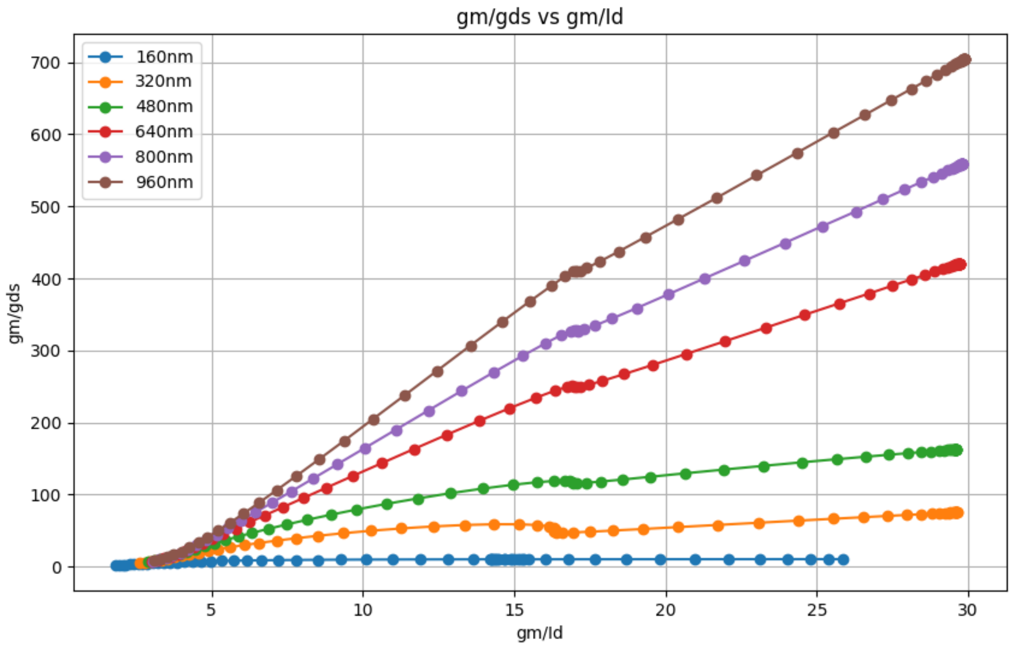

Figure 35: gm*ro vs gm/Id for PMOS at Vsd = 0.5 V

– **ft vs gm/Id**

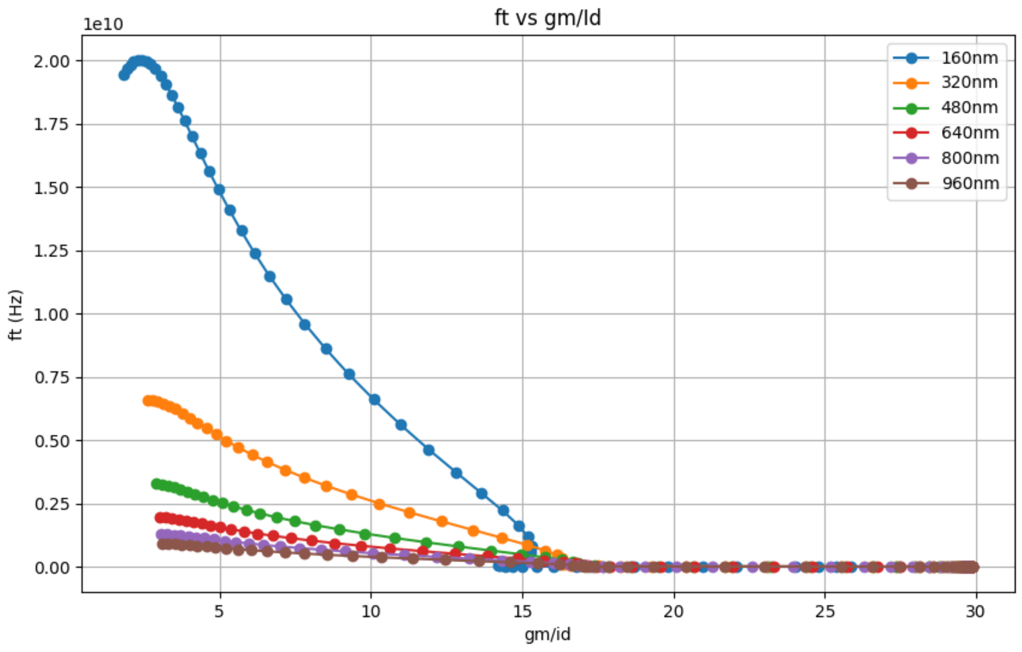

Figure 36: ft vs gm/Id for PMOS at Vsd = 0.5 V

– **Id/W vs gm/Id**

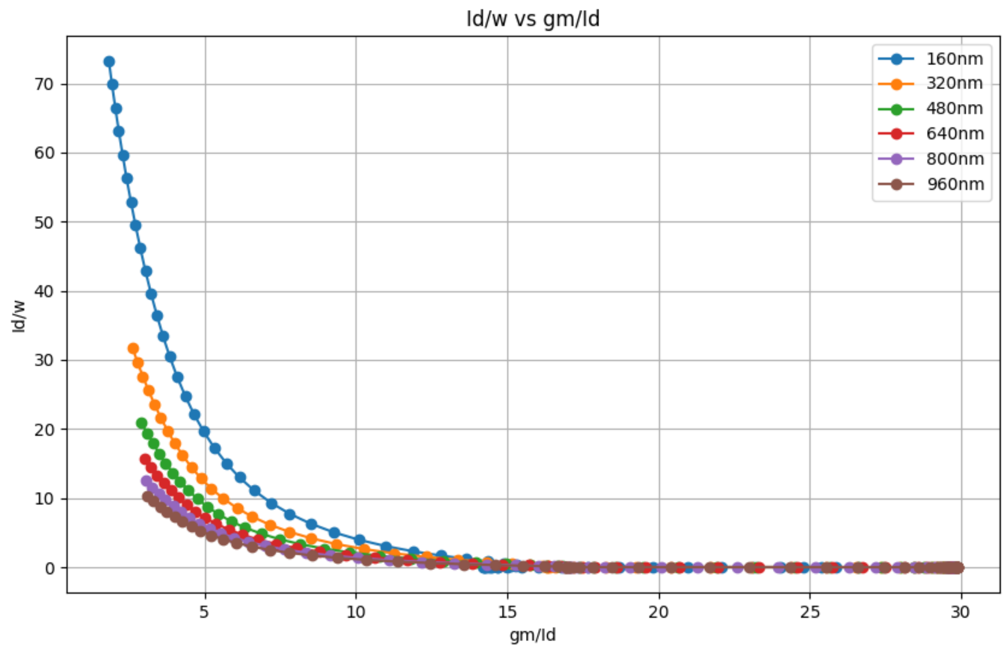

Figure 37: Id/W vs gm/Id for PMOS at Vsd = 0.5 V

4. **Vsd: 0.7 V**
   - **gm*ro vs gm/Id**

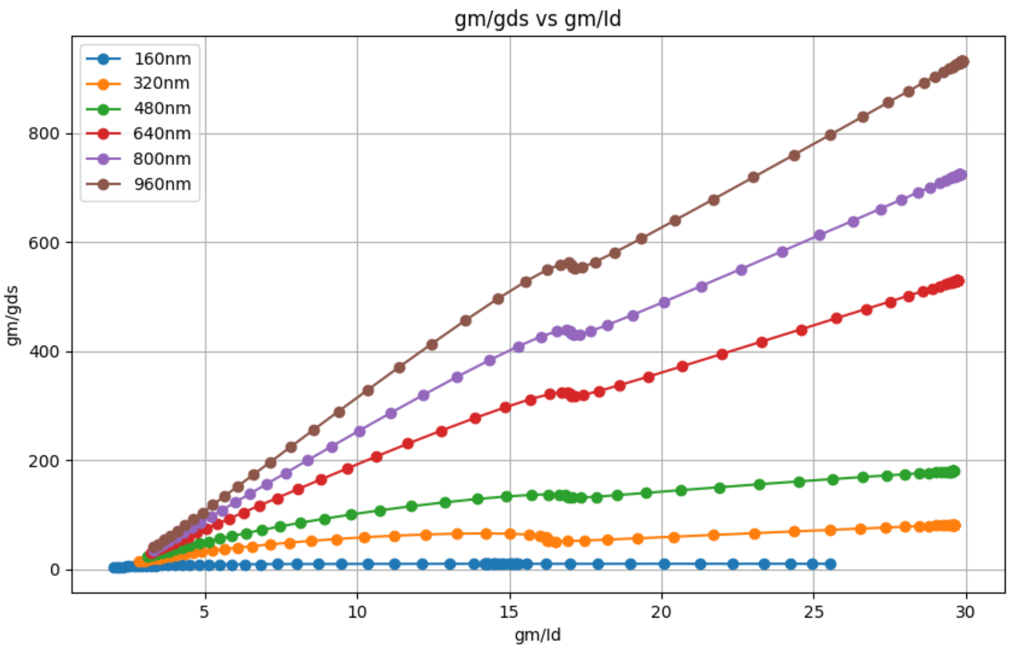

Figure 38: gm*ro vs gm/Id for PMOS at Vsd = 0.7 V

   - **ft vs gm/Id**

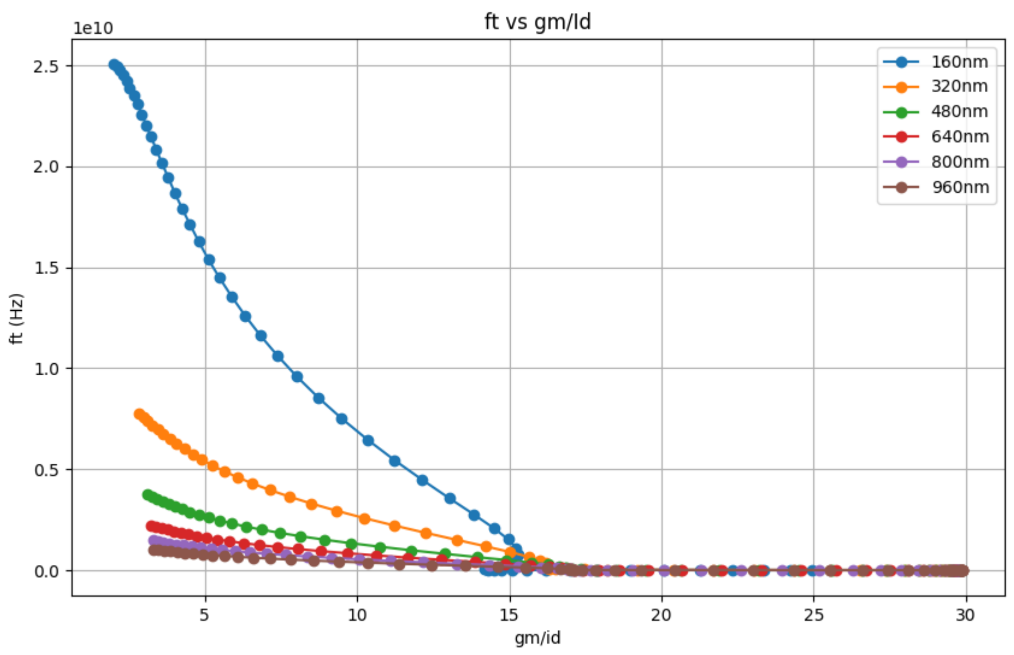

Figure 39: ft vs gm/Id for PMOS at Vsd = 0.7 V

– **Id/W vs gm/Id**

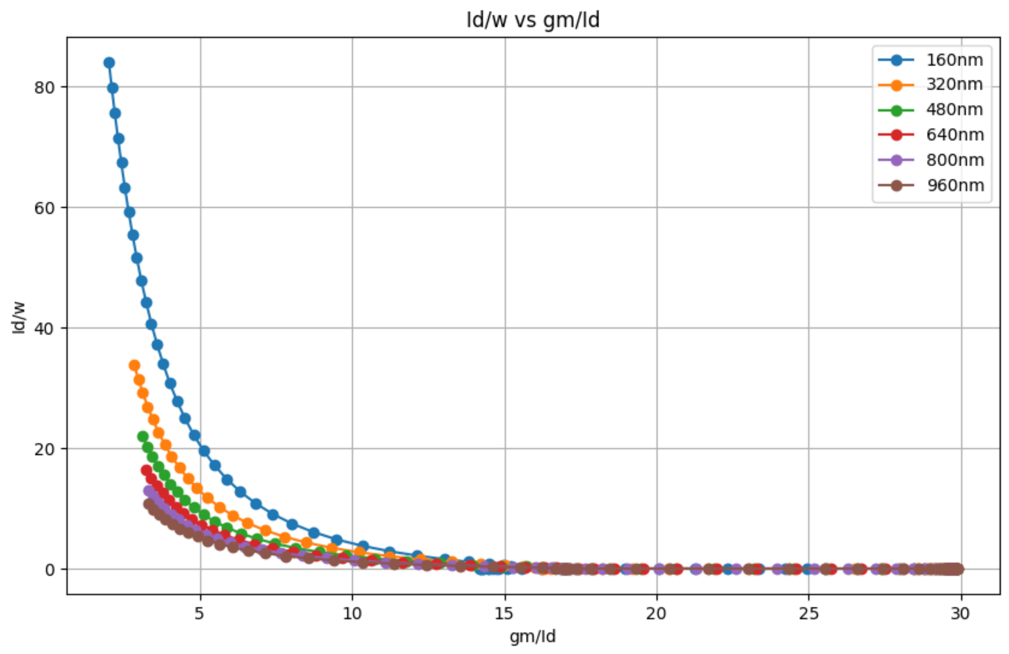

Figure 40: Id/W vs gm/Id for PMOS at Vsd = 0.7 V

Comparison with Square Law Model for SKYWATER130

Consider Skywater NMOS Model with threshold voltage of 0.35 V. Following are the parameters for the comparison:

- Length: 320nm
- $V_D = 0.3V$

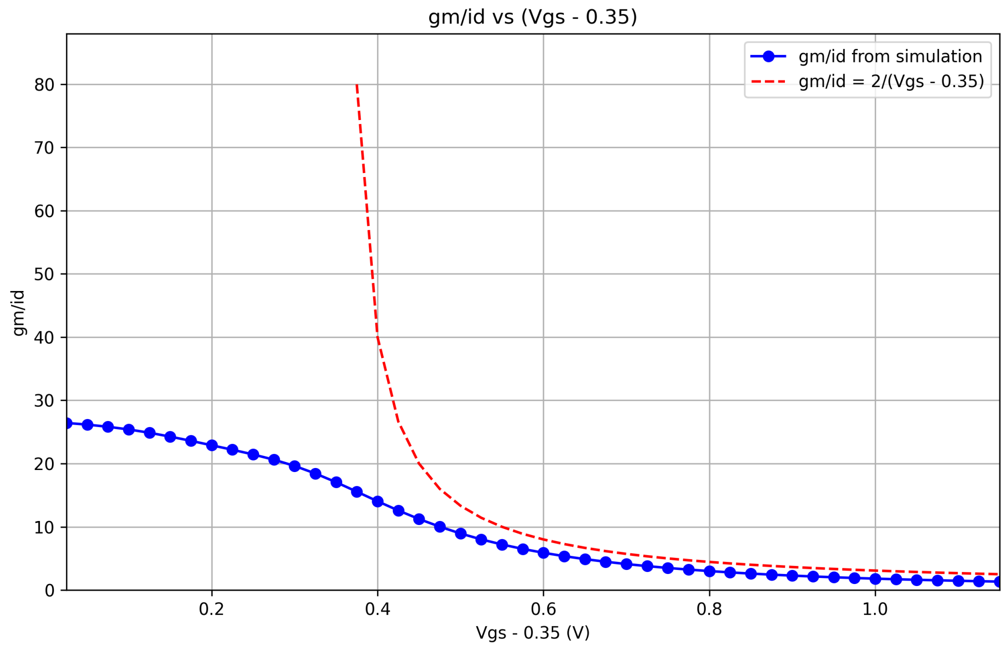

Figure 41: Comparison of g_m/I_D vs. V_{ov}: Simulation vs. Square Law.

Comparison with Square Law Model for SKYWATER130

Consider Skywater PMOS Model with threshold voltage of 0.23 V. Following are the parameters for the comparison:

- Length: 320nm
- $V_s = 0.3V$

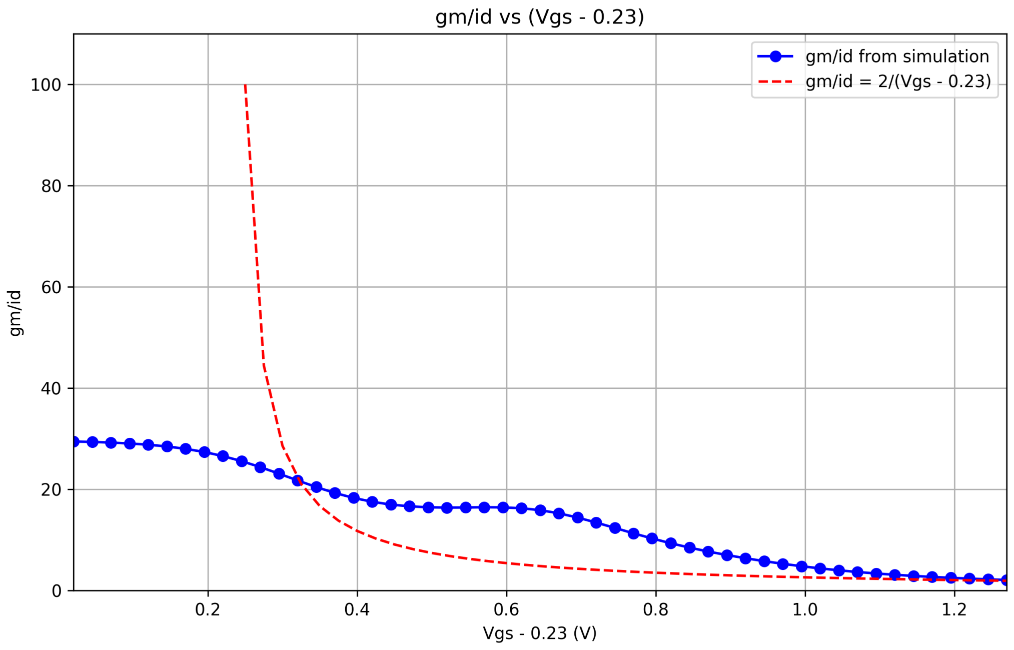

Figure 42: Comparison of g_m/I_D vs. V_{ov}: Simulation vs. Square Law.

Techplots Comparison for Skywater130 and IHP PDK (for NMOS)

Consider the three techplots below for 480nm and Vds = 0.1 V and observe the values for the same paramters which are applied.

– Intrinsic Gain vs gm/Id

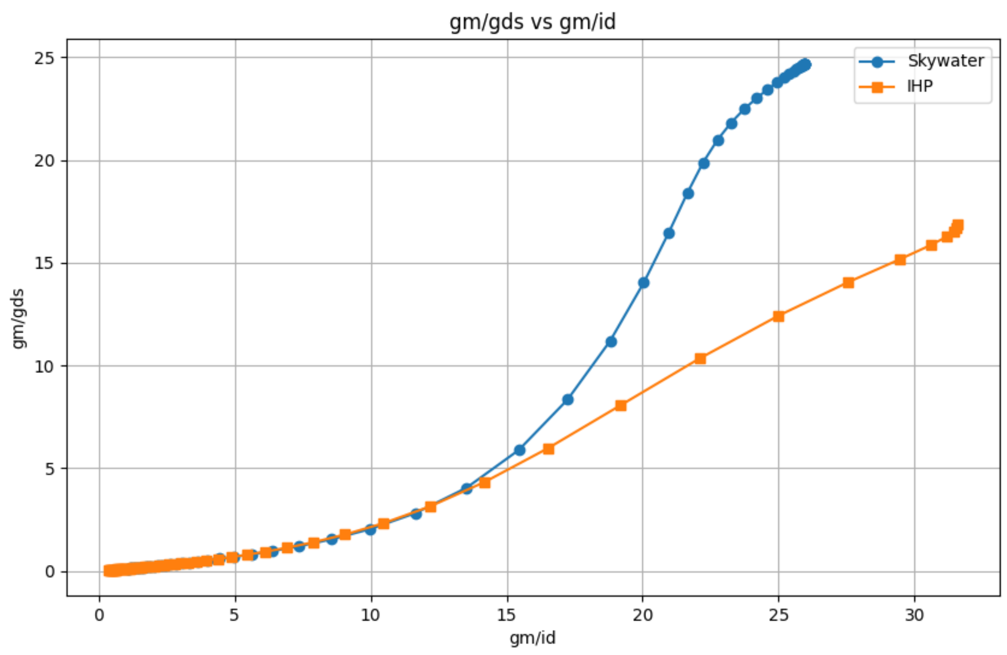

Figure 43: Comparison for Intrinsic Gain

– Transition Frequency vs gm/Id

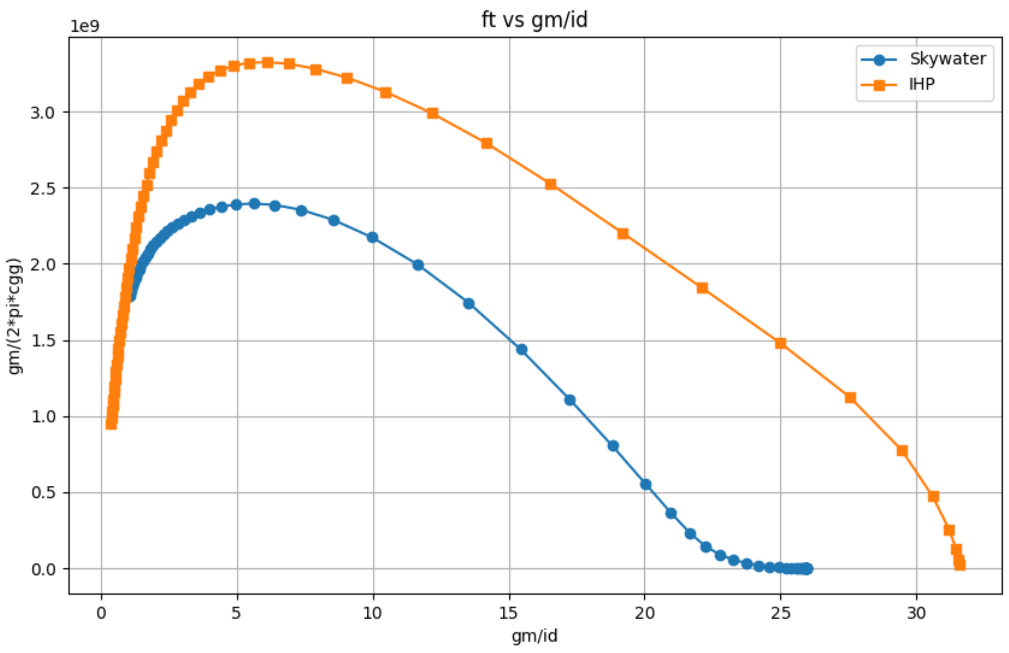

Figure 44: Comparison for Transition Frequency

– Id/W vs gm/Id

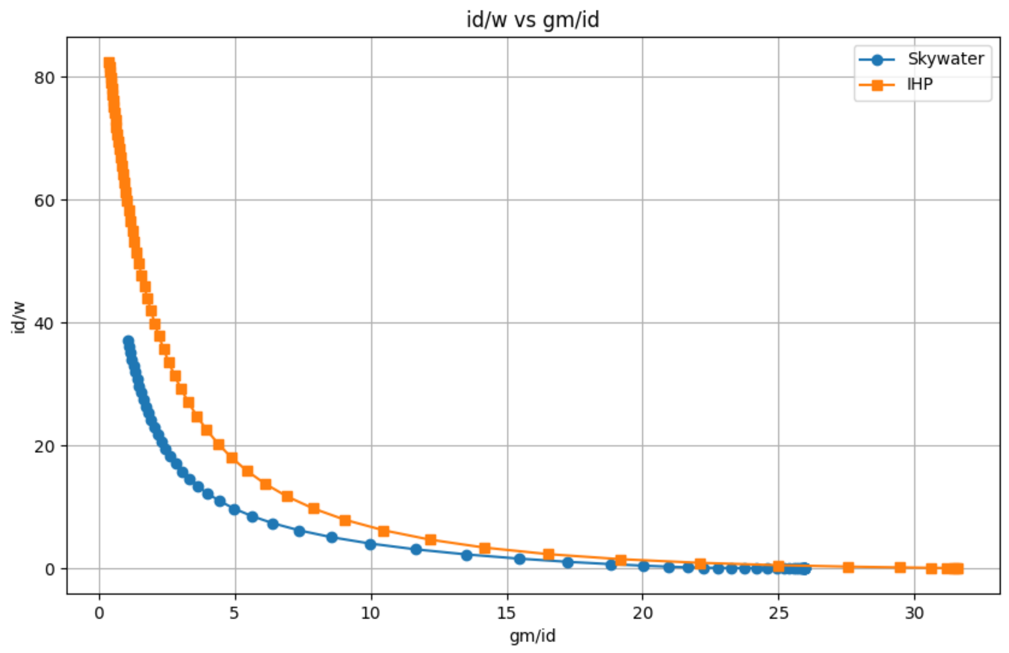

Figure 45: Comparison for Id/W

Key Differences in Figures of Merit for SkyWater and IHP

- It appears that for weak inversion, there exists a discrepancy between the plots for Skywater130's and IHP's intrinsic gain. Although for strong inversion, the plots have identical values.
- The plot for Id/W seems to be identical for the most part, especially for weak inversion and subthreshold.
- The transition frequencies for Skywater130 and IHP for a particular gm/Id are different all throughout the Vov range.

6. Challenges

- **Challenges related to installation -**
  Initially, if we find resources to download and install PDKs for IHP, we reach out to GitHub at https://github.com/IHP-GmbH/IHP-Open-PDK, provided by IHP itself. But as of now, there is a file that is missing over this GitHub, due to which we can't install these PDKs. The correct GitHub repo as of now to download IHP PDK is https://github.com/iic-jku/IIC-OSIC-TOOLS.
- **Challenges related to obtaining parameters -**
  After writing the testbench, click on the netlist and then on simulate (on the top right). You'll be able to see "view data." In that "Direct Linear Solver" should run multiple times, this will generate enough values for us to plot a graph.

- **Challenges related to testbench -**
  Sometimes composing l vec (to get parameters for different lengths) doesn't work for every PDK. In the case of Skywater 130nm, we need to specify one length at a time to obtain the parameters for it. Challenges regarding generating tech plots for PMOS from IHP PDK The gm parameter contains negative values, which can create distinct trends in the plot, leading to two lines. We're looking ahead to solving this problem.
- **Challenges regarding generating tech plots for PMOS from IHP PDK -**
  The gm parameter contains negative values, which can create distinct trends in the plot, leading to two lines. We're looking ahead to solving this problem.

7. Future Proceedings

- Evaluate and Compare the results for PMOS in IHP
- Compare the PMOS plots for Skywater and IHP
- Compare the PMOS Plots v/s square law in IHP PDK.
- Multi-Node Comparison: Compare the figures of merit for 180 nm, 130 nm, and 45 nm nodes
- LDO in IHP: Design the LDO in just the spreadsheet using techplots.

8. Upcoming Conferences

- **2025 IEEE PowerTech**
  - **Conference Start Date:** 29 June 2025
  - **Conference End Date:** 3 July 2025
  - **Location:** Kiel, Germany
  - **Submission Due Date:** 10 January 2025
- **ECCE Europe 2025**
  - **Final deadline for submission of provisional full paper:** 25 February 2025
  - **Notification of acceptance** 25 April 2025
  - **Location:** Birmingham, UK
  - **Submission of final full paper:** 10 June 2025
- **IEEE European Solid-State Electronics Research Conference**
  - **Dates:** 8-11 Sept 2025
  - **Location:** Munich, Germany
  - **Submission of final full paper:** 4 April 2025

9. References

– This article by Herald Pretl and Michael Koefinger

– IIC-OSIC-TOOLS Repository

– This thread is particularly useful to understand the non-physical behaviour of the MOSFETs.

– IHP-Open-Source PDK repository.

– Google's Skywater PDK repository.