6.2500 ARM Core EDP Simulator Usage

Your class project is to design a MOSFET that optimizes the energy-delay product (EDP) of an ARM microprocessor core.

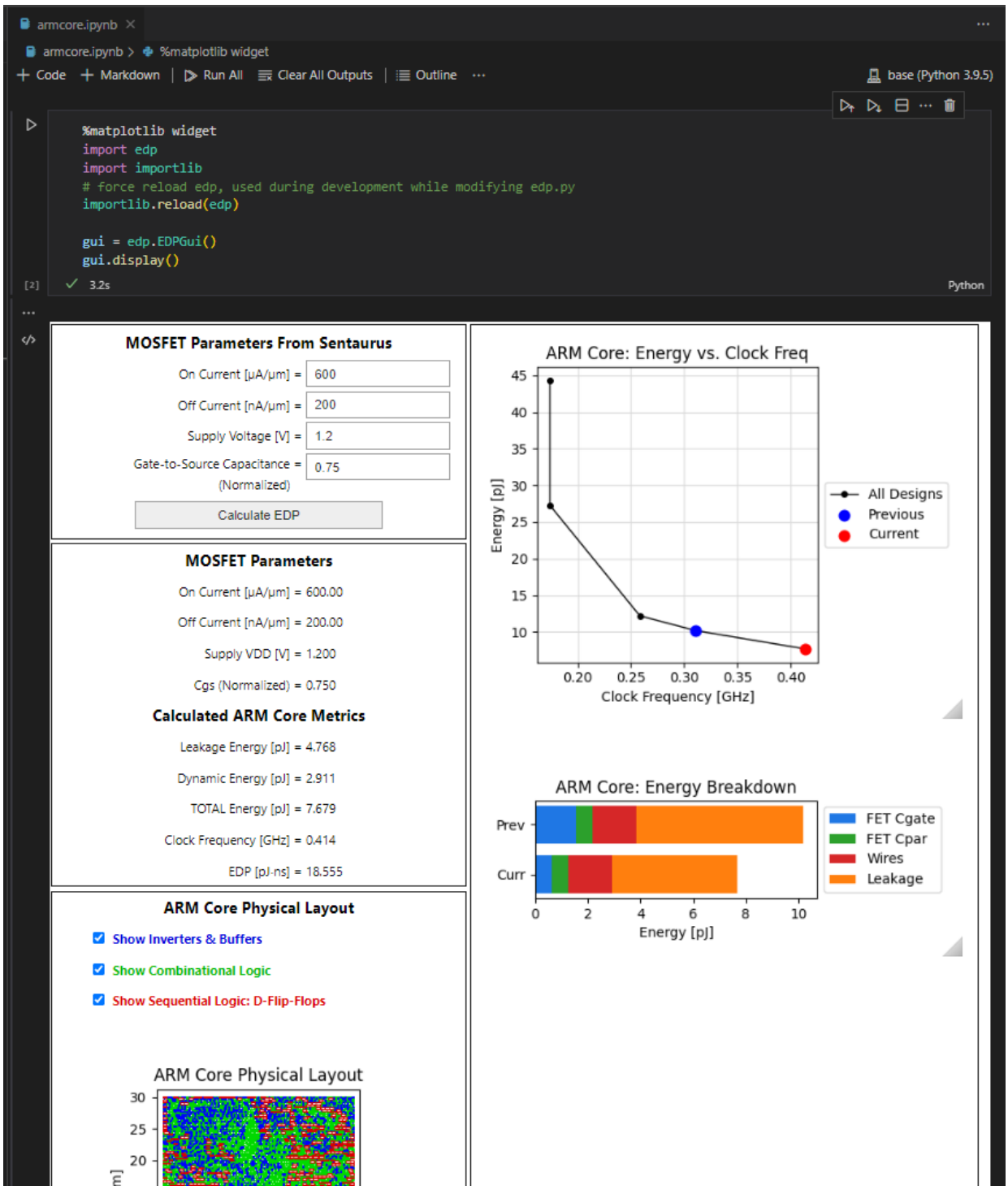
Your goal is to engineer key MOSFET performance parameters:

- on-current I_ON
- off-current I_OFF
- supply voltage V_DD
- normalized gate-to-source capacitance (relative to baseline)

Your output is the ARM core's delay, energy, and edp metrics. We provide you with a layout of an ARM core processor and functions to simulate the energy and delay of the ARM core. We provide three ways of doing the ARM core simulation using either Python or MATLAB:

1. Python jupyter notebook GUI

Default tool is a python based ARM Core EDP simulator that is built as a Jupyter notebook gui, here's what it looks like:



Installation

First install a latest Python version: https://www.python.org/downloads/

Next, download the project python files in `PYTHON_ARM_CORE_EDP.zip`. Unzip the files into a folder. Open a terminal (on Mac or Linux) or powershell (on Windows) **inside the folder** and install the python package requirements using pip:

```
pip install -r requirements.txt
```

If above does not work, try

```
python -m pip install -r requirements.txt
```

On linux or mac, it may also be

```
python3 -m pip install -r requirements.txt
```

Here is another general tutorial on installing python and VS code (below):
https://code.visualstudio.com/docs/python/python-tutorial

Running ARM Core GUI Using VS Code

The popular code editor VS Code is able to run Jupyter notebooks locally and is what we recommend for running the python ARM Core EDP simulator:

1. Download VS Code: https://code.visualstudio.com/download
2. Install **Python** and **Jupyter** extensions
3. Open the `armcore.ipynb` jupyter notebook in VS code and hit "Run All". This will create the GUI tool. You can then enter MOSFET parameters from Sentaurus into the boxes in the top left. Hit the "Calculate" button to update the EDP Simulation.
4. If VS Code asks for which "Python kernel" to use, select "base". This will use the default python installation and packages. If you instead want to use a python virtual environment (venv) to isolate your packages, you will need to figure that out yourself.
5. If VS Code asks for "Widgets requires 3rd party download", say Okay. The 3rd party download is loading javascript files for styling and interaction in the Jupyter notebook gui widgets.

Known Issues

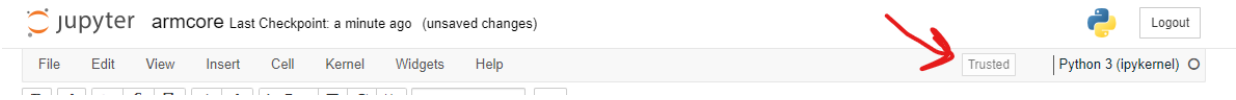
- If you installed VS Code then installed packages afterwards, if you had VS Code open at the time, the GUI may not output. Try clearing all, then close and reopen VS Code.
- On mac, python 2 is installed by default. Make sure you install a **python 3** version for the code to work.

Manual Jupyter Notebook in Web Browser

You can use any alternative for Jupyter notebook. If you are manually running a Jupyter notebook using the `jupyter` command and opening the GUI in a web browser,

```
jupyter notebook armcore.ipynb
```

Make sure you have selected "Trusted" for the notebook in the top right corner, otherwise the notebook will not display the GUI widgets properly.



ARM Core GUI Usage

There are five main components in the GUI:

1. **MOSFET Parameters from Sentaurus**: (Left-top) Enter your derived MOSFET parameters from Sentaurus current-voltage ID-VGS plot here. Hit "Calculate EDP" to run the ARM Core simulation and update the derived metrics.
2. **MOSFET Parameters/Calculated ARM Core Metrics**: (Left-middle) After pressing "Calculate EDP", the parameters used and calculated energy and delay summary will be printed here. These are the final numbers you will submit for your project.
3. **ARM Core: Energy vs. Clock Freq**: (Right-top) This plots clock frequency versus total energy. Note

   ```
   EDP = energy * delay = energy * 1/frequency
   ```

   In general, you want to be as "down-and-right" on the plot as possible (higher frequency and lower energy).
4. **ARM Core: Energy Breakdown**: (Right-bottom) This breaks down the components of total energy of your current and previous design. This is to help you visualize what is dominating your energy consumption, and how your transistor optimizations affected each energy component. We hope this helps you target which energy consumption you can optimize. Here is what each component means:
   - **FET Cgate**: This is the intrinsic `Cgate*VDD^2` energy needed to switch your MOSFET gates. This is "intrinsic" in that it is the core gate capacitance needed. You can only optimize this by reducing `VDD`.
   - **FET Cpar**: This is parasitic capacitances from the spacer capacitance (between gate and source/drain metal contacts) and the junction capacitances from the PN junction depletion capacitance inside the N++ contact implant and the P substrate.
   - **Wires**: These are parasitic wire RC energy.
   - **Leakage**: This is the off-state leakage energy. Note this is `Eleakage = Pleakage * Tclk`, so you can reduce this both by running faster (to reduce proportion of leakage) and/or reducing leakage power by reducing off current.
5. **ARM Core Physical Layout**: (Left-bottom) This is just a cool visualization of the real physical circuit layout. But it is not needed in any of your calculations.

The Jupyter notebook GUI will automatically save and reload your results history in the plots, so you can track your optimization history path for your report. If you want to clear the plot, simply delete the created `results.csv` file that will be created in the folder each time you run "Calculate". You can also edit the csv file to remove points from the plot as needed. If you delete or edit the `results.csv` file, hit "Run All" again at the top of the tab to reload the Jupyter notebook.

2. Python command line or as an external function

If you don't want to use the python jupyter notebook gui, you can create the ARM core simulator and calculate the edp directly.

Install the python requirements using pip just like above. But instead of the jupyter notebook, see file `PYTHON_ARM_CORE_EDP/example_no_gui.py` for how to use the simulator directly. The rough code will look like

```
from edp import ARMCore

# Create a re-usable instance of the ARMCore simulator. This will load
# the ARM core circuit model and layout into memory and do some
# pre-calculations.
armcore = ARMCore()

# we can re-use this instance to calculate the delay and energy
# simple calculation:
results = armcore.calculate_energy_delay(
    i_on=600,    # in uA/um
    i_off=400,   # in nA/um
    v_dd=1.8,    # supply voltage
    c_gs=1.0,    # normalized to 1
)

# print results dict
print("RESULTS:")
for key, value in results.items():
    print(f"{key} = {value}")
```

You can run this as any other python script,

```
python example_no_gui.py
```

You can also use this to explore the design space, but remember the actual on and off currents you submit must be physically achievable and derived from your Sentaurus simulations!

3. MATLAB GUI (Original tool)

If you have MATLAB and want to use MATLAB, we have our old original MATLAB GUI tool for calculating the ARM core EDP. Unzip the `MATLAB_ARM_CORE_EDP.zip` and read the `user_guide-ARM_CORE_EDP.pptx` for tutorial on its usage. Overall it is very similar to the Python GUI.