import streamlit as st

import numpy as np

import matplotlib.pyplot as plt

from scipy import signal

import math

# --- Constants ---

CLIPPING\_LIMIT = 15.0 # Define the clipping limit for output voltage

# --- Helper Functions for Waveform Generation and Simulation ---

def get\_actual\_frequency(freq\_val, unit):

"""Converts frequency value based on selected unit."""

if unit == "kHz":

return freq\_val \* 1e3

elif unit == "MHz":

return freq\_val \* 1e6

else: # Hz

return freq\_val

def generate\_waveform(amp, freq, wave\_type, num\_cycles=3):

"""Generates the specified waveform."""

sampling\_rate = 1000000

if freq == 0:

duration = 0.01

t = np.linspace(0, duration \* num\_cycles, int(sampling\_rate \* duration \* num\_cycles), endpoint=False)

y = np.zeros\_like(t)

else:

period = 1 / freq

total\_duration = period \* num\_cycles

num\_points = int(sampling\_rate \* total\_duration)

if num\_points < 1000:

num\_points = 1000

t = np.linspace(0, total\_duration, num\_points, endpoint=False)

if wave\_type == "Sine wave":

y = amp \* np.sin(2 \* np.pi \* freq \* t)

elif wave\_type == "Cosine wave":

y = amp \* np.cos(2 \* np.pi \* freq \* t)

elif wave\_type == "Triangular wave":

y = amp \* signal.sawtooth(2 \* np.pi \* freq \* t, width=0.5)

elif wave\_type == "Square wave":

y = amp \* signal.square(2 \* np.pi \* freq \* t)

else: # Default or no selection

y = np.zeros\_like(t)

return y, t, amp, total\_duration, freq

def get\_amplifier\_name(amp\_type\_value):

"""Returns human-readable amplifier name."""

if amp\_type\_value == "Inverting Amplifier":

return "Inverting"

elif amp\_type\_value == "Non-Inverting Amplifier":

return "Non-Inverting"

elif amp\_type\_value == "Buffer Amplifier":

return "Buffer"

return "N/A"

def calculate\_amplifier\_output(y\_input, amp\_input, R1\_kohm, Rf\_kohm, amplifier\_type\_name):

"""Calculates amplifier output based on type and resistances."""

R1\_val = R1\_kohm \* 1000

Rf\_val = Rf\_kohm \* 1000

y\_output = np.zeros\_like(y\_input)

output\_amplitude = 0

phase\_diff\_deg = 0

if amplifier\_type\_name == "Inverting Amplifier":

if R1\_val != 0:

gain = -(Rf\_val / R1\_val)

y\_output = gain \* y\_input

output\_amplitude = abs(gain) \* amp\_input

phase\_diff\_deg = 180 if amp\_input != 0 else 0

else:

output\_amplitude = 0

y\_output = np.zeros\_like(y\_input)

phase\_diff\_deg = 0

elif amplifier\_type\_name == "Non-Inverting Amplifier":

if R1\_val != 0:

gain = 1 + (Rf\_val / R1\_val)

y\_output = gain \* y\_input

output\_amplitude = gain \* amp\_input

phase\_diff\_deg = 0

else: # R1 = 0, behaves as buffer

gain = 1

y\_output = gain \* y\_input

output\_amplitude = gain \* amp\_input

phase\_diff\_deg = 0

elif amplifier\_type\_name == "Buffer Amplifier":

gain = 1

y\_output = gain \* y\_input

output\_amplitude = amp\_input

phase\_diff\_deg = 0

# --- Output Clipping Logic ---

y\_output = np.clip(y\_output, -CLIPPING\_LIMIT, CLIPPING\_LIMIT)

if np.all(y\_output == 0):

output\_amplitude = 0

else:

output\_amplitude = np.max(np.abs(y\_output))

return y\_output, output\_amplitude, phase\_diff\_deg

# --- Main App Configuration ---

st.set\_page\_config(

page\_title="Electronics Lab Simulator",

layout="wide",

initial\_sidebar\_state="collapsed"

)

# Initialize session state for page navigation

if 'page' not in st.session\_state:

st.session\_state.page = 'home'

# --- Header Section ---

col\_text, col\_logo = st.columns([4, 1])

with col\_text:

st.title("SRM INSTITUTE OF SCIENCE AND TECHNOLOGY")

st.subheader("Department of Electronics and Communication Engineering")

with col\_logo:

st.image("image\_a2e0d8.png", width=150)

# --- Navigation functions ---

def navigate\_to(page\_name):

"""Sets the page in session state to navigate to."""

st.session\_state.page = page\_name

def show\_home\_page():

"""Displays the main page with clickable experiment cards."""

st.title("Welcome to the Electronics Lab Simulator!")

st.write("Click on an experiment to start the simulation.")

experiments = {

"Basic Op-Amp Simulator": "A basic simulation for common op-amp configurations.",

"Integrator/Differentiator": "Simulate the behavior of op-amp based integrator and differentiator circuits with various input waveforms.",

"Precision Rectifier": "Explore half-wave and full-wave precision rectifier circuits.",

"Schmitt Trigger": "Simulate the hysteresis behavior of a Schmitt Trigger circuit.",

"Active Wave Shaping": "Experiment with active clipper and clamper circuits.",

"RC Phase Shift Oscillator": "Simulate the frequency and component requirements for an RC phase shift oscillator.",

"Wien Bridge Oscillator": "Simulate the frequency and component requirements for a Wien Bridge oscillator.",

"Square Wave Generator": "Design and simulate an op-amp based square wave generator (astable multivibrator).",

"Active Filter": "Analyze the frequency response of active lowpass and highpass filters."

}

# Create a grid of experiment cards using columns

cols = st.columns(3)

col\_index = 0

for exp\_title, exp\_desc in experiments.items():

with cols[col\_index]:

# Use st.expander to create a box-like, collapsible card

with st.expander(exp\_title):

st.write(exp\_desc)

# Use a button to trigger the navigation

if st.button("Launch Simulation", key=f"launch\_{exp\_title}", use\_container\_width=True):

navigate\_to(exp\_title)

col\_index = (col\_index + 1) % 3

st.markdown("---")

st.write("Developed by Dr. Damodar Panigrahy, Assistant Professor, Department of ECE, SRMIST, Kattankulathur.")

# --- Experiment Pages ---

def show\_basic\_op\_amp\_simulator():

"""Content for the Basic Op-Amp Simulator page with simulation logic."""

st.header("Basic Amplifier Simulator")

st.button("Return to Home", on\_click=navigate\_to, args=('home',))

st.markdown("---")

# --- Create tabs here ---

prelab\_tab, theory\_tab, simulation\_tab, postlab\_tab, feedback\_tab = st.tabs([

"Prelab", "Theory", "Simulation", "Postlab", "Feedback"

])

with prelab\_tab:

st.subheader("Prelab Questions")

st.write("Before starting the simulation, answer the following questions to test your understanding.")

st.markdown("""

1. What is an ideal Op-Amp, and what are its key characteristics?

2. Derive the gain equation for a non-inverting amplifier.

3. Explain the function of the feedback resistor ($R\_f$) in an inverting amplifier circuit.

4. What is the purpose of a buffer amplifier?

5. How does the output voltage of an Op-Amp behave when it reaches its saturation limits?

""")

st.text\_area("Your answers here...", height=200)

st.button("Submit Answers")

with theory\_tab:

st.subheader("Circuit Theory")

st.write("This section provides a brief overview of the theoretical concepts behind the circuits you will be simulating.")

st.markdown("#### The Operational Amplifier (Op-Amp)")

st.write("An operational amplifier is a DC-coupled high-gain electronic voltage amplifier with a differential input and, usually, a single-ended output. A key characteristic of the ideal Op-Amp is that it has infinite input impedance and zero output impedance.")

st.markdown("#### Inverting Amplifier")

st.write("In this configuration, the input signal is applied to the inverting input terminal. The output voltage is out of phase with the input and its gain is determined by the ratio of the feedback resistor ($R\_f$) to the input resistor ($R\_1$).")

st.markdown("[Image of an inverting amplifier circuit]")

st.latex(r"V\_{out} = -\left(\frac{R\_f}{R\_1}\right) V\_{in}")

st.markdown("#### Non-Inverting Amplifier")

st.write("Here, the input signal is applied to the non-inverting input terminal. The output voltage is in phase with the input and its gain is given by the formula:")

st.markdown("[Image of a non-inverting amplifier circuit]")

st.latex(r"V\_{out} = \left(1 + \frac{R\_f}{R\_1}\right) V\_{in}")

st.markdown("#### Buffer Amplifier (Voltage Follower)")

st.write("A buffer amplifier is a non-inverting amplifier with a gain of 1. It is used to isolate one stage of a circuit from another, providing high input impedance and low output impedance.")

st.markdown("")

st.latex(r"V\_{out} = V\_{in}")

with simulation\_tab:

# Create columns for layout

col1, col2, col3 = st.columns([1, 1, 2]) # Function Generator, Amplifier, CRO Displays

with col1:

st.header("Function Generator")

wave\_type = st.radio(

"Select Waveform:",

("Sine wave", "Cosine wave", "Triangular wave", "Square wave", "None"),

index=0, # Default to Sine wave

key="wave\_type\_radio"

)

amplitude = st.slider(

"Amplitude (V)",

min\_value=0.0, max\_value=5.0, value=1.0, step=0.01,

format="%.2f V",

key="amplitude\_slider"

)

st.write("Frequency")

freq\_col1, freq\_col2 = st.columns([2, 1])

with freq\_col1:

frequency\_value = st.slider(

"Frequency Value",

min\_value=0.0, max\_value=1100.0, value=100.0, step=0.1,

label\_visibility="collapsed", # Hide default label to combine with units

key="frequency\_slider"

)

with freq\_col2:

freq\_unit = st.radio(

"Unit",

("Hz", "kHz", "MHz"),

index=0, # Default to Hz

horizontal=True,

label\_visibility="collapsed", # Hide default label

key="freq\_unit\_radio"

)

actual\_frequency = get\_actual\_frequency(frequency\_value, freq\_unit)

with col2:

st.header("Amplifier Settings")

amplifier\_type = st.radio(

"Select Amplifier Type:",

("Inverting Amplifier", "Non-Inverting Amplifier", "Buffer Amplifier", "None"),

index=0, # Default to Inverting Amplifier

key="amp\_type\_radio"

)

r1\_kohm = st.number\_input(

"Input Resistance ($R\_1$) (kΩ)",

min\_value=0.01,

value=10.0,

step=0.1,

format="%.2f",

key="r1\_input"

)

rf\_kohm = st.number\_input(

"Feedback Resistance ($R\_f$) (kΩ)",

min\_value=0.01,

value=100.0,

step=0.1,

format="%.2f",

key="rf\_input"

)

# Use a state variable for the simulation results table

if 'simulation\_results' not in st.session\_state:

st.session\_state.simulation\_results = []

if 'row\_id\_counter' not in st.session\_state:

st.session\_state.row\_id\_counter = 0

# --- Simulation Logic and Plotting (triggered when inputs change) ---

y\_input, t, amp\_input, total\_duration, input\_freq = generate\_waveform(amplitude, actual\_frequency, wave\_type)

y\_output, output\_amplitude, phase\_diff\_deg = calculate\_amplifier\_output(

y\_input, amp\_input, r1\_kohm, rf\_kohm, amplifier\_type

)

with col3:

st.header("CRO Displays")

st.subheader("Circuit Diagram")

if amplifier\_type == "Inverting Amplifier":

st.image("images/invertingamplifier.png", caption="Inverting Amplifier Circuit", use\_column\_width=True)

elif amplifier\_type == "Non-Inverting Amplifier":

st.image("images/Noninvertingamplifier.png", caption="Non-Inverting Amplifier Circuit", use\_column\_width=True)

elif amplifier\_type == "Buffer Amplifier":

st.image("images/voltagefollower.png", caption="Buffer Amplifier (Voltage Follower) Circuit", use\_column\_width=True)

else:

st.info("Select an amplifier type to display its circuit diagram.")

st.markdown("---")

st.subheader("CRO Waveforms")

# Plot 1: Input Signal

fig1, ax1 = plt.subplots(figsize=(6, 3))

ax1.plot(t, y\_input, color='lime')

ax1.set\_facecolor("black")

ax1.axhline(0, color='gray', linewidth=0.5)

ax1.axvline(0, color='gray', linewidth=0.5)

ax1.set\_ylim(-amplitude \* 1.5 if amplitude > 0 else -1, amplitude \* 1.5 if amplitude > 0 else 1)

ax1.set\_xlim(0, total\_duration)

ax1.tick\_params(axis='x', colors='white')

ax1.tick\_params(axis='y', colors='white')

ax1.set\_title("Input Waveform", color='black')

if amplitude != 0:

ax1.text(0.02, 0.95, f'Amplitude: {amplitude:.2f} V', transform=ax1.transAxes,

fontsize=10, color='white', verticalalignment='top')

st.pyplot(fig1)

plt.close(fig1)

# Plot 2: Output Signal

fig2, ax2 = plt.subplots(figsize=(6, 3))

ax2.plot(t, y\_output, color='cyan')

ax2.set\_facecolor("black")

ax2.axhline(0, color='gray', linewidth=0.5)

ax2.axvline(0, color='gray', linewidth=0.5)

plot\_ylim = max(output\_amplitude \* 1.2, 1.0)

ax2.set\_ylim(-plot\_ylim, plot\_ylim)

ax2.set\_xlim(0, total\_duration)

ax2.tick\_params(axis='x', colors='white')

ax2.tick\_params(axis='y', colors='white')

ax2.set\_title("Output Waveform", color='black')

amplitude\_display\_text = f'Amplitude: {output\_amplitude:.2f} V'

if abs(output\_amplitude - CLIPPING\_LIMIT) < 0.01 and amplitude > 0:

amplitude\_display\_text += ' (Clipped)'

elif output\_amplitude == 0 and amp\_input != 0 and amplifier\_type != "None":

amplitude\_display\_text += ' (No Output)'

ax2.text(0.02, 0.95, amplitude\_display\_text, transform=ax2.transAxes,

fontsize=10, color='white', verticalalignment='top')

st.pyplot(fig2)

plt.close(fig2)

# Plot 3: Combined View

fig\_combined, ax\_combined = plt.subplots(figsize=(6, 4))

ax\_combined.plot(t, y\_input, color='lime', label='Input (Ch 1)')

ax\_combined.plot(t, y\_output, color='cyan', label='Output (Ch 2)')

ax\_combined.set\_facecolor("black")

ax\_combined.axhline(0, color='gray', linewidth=0.5)

ax\_combined.axvline(0, color='gray', linewidth=0.5)

max\_combined\_amp = max(amplitude \* 1.5, plot\_ylim) if amplitude > 0 else max(1.0, plot\_ylim)

ax\_combined.set\_ylim(-max\_combined\_amp, max\_combined\_amp)

ax\_combined.set\_xlim(0, total\_duration)

ax\_combined.tick\_params(axis='x', colors='white')

ax\_combined.tick\_params(axis='y', colors='white')

ax\_combined.set\_title("Combined Waveform", color='black')

ax\_combined.legend(loc='upper right', facecolor='darkgray', edgecolor='white')

st.pyplot(fig\_combined)

plt.close(fig\_combined)

# --- Simulation Results Table ---

st.markdown("---") # Horizontal line for separation

st.header("Simulation Results")

# Button to log current simulation

if st.button("Log Current Simulation"):

st.session\_state.row\_id\_counter += 1

new\_entry = {

"#": st.session\_state.row\_id\_counter,

"Amplifier Type": get\_amplifier\_name(amplifier\_type),

"R1 (kΩ)": f"{r1\_kohm:.1f}",

"Rf (kΩ)": f"{rf\_kohm:.1f}",

"Input Amp (V)": f"{amp\_input:.2f}",

"Input Freq (Hz)": f"{input\_freq:.1f}",

"Output Amp (V)": f"{output\_amplitude:.2f}",

"Output Freq (Hz)": f"{input\_freq:.1f}",

"Phase Diff (deg)": f"{phase\_diff\_deg:.1f}"

}

st.session\_state.simulation\_results.append(new\_entry)

# Display the table

if st.session\_state.simulation\_results:

st.dataframe(st.session\_state.simulation\_results, hide\_index=True)

else:

st.info("No simulations logged yet. Adjust parameters and click 'Log Current Simulation'.")

# Button to clear table

if st.button("Clear Log"):

st.session\_state.simulation\_results = []

st.session\_state.row\_id\_counter = 0

st.rerun()

with postlab\_tab:

st.subheader("Postlab Report")

st.write("Complete the following report based on your simulation results.")

st.text\_area("Observations and key findings...", height=200)

st.text\_area("Conclusion...", height=100)

st.button("Generate PDF Report")

with feedback\_tab:

st.subheader("Provide Feedback")

st.write("We would love to hear your thoughts on this simulator.")

st.text\_input("Your Name (Optional)")

st.slider("How would you rate this simulator?", 1, 5)

st.text\_area("Your comments...")

st.button("Submit Feedback")

def show\_integrator\_differentiator():

"""Content for the Integrator/Differentiator page with simulation logic."""

st.header("Integrator/Differentiator")

st.button("Return to Home", on\_click=navigate\_to, args=('home',))

st.sidebar.header("Circuit Configuration")

circuit\_type = st.sidebar.selectbox("Choose Circuit Type", ["Integrator", "Differentiator"])

st.sidebar.header("Component Values")

R = st.sidebar.number\_input("Resistor ($R$, $\\Omega$)", value=10000, min\_value=1)

C = st.sidebar.number\_input("Capacitor ($C$, $\\mu$F)", value=1, min\_value=0.01, format="%.2f") \* 1e-6

st.sidebar.header("Input Signal Properties (Square Wave)")

vin\_amplitude = st.sidebar.slider("Input Voltage Amplitude (V)", min\_value=1.0, max\_value=10.0, value=5.0, step=0.1)

frequency = st.sidebar.slider("Frequency (Hz)", min\_value=10.0, max\_value=1000.0, value=100.0, step=10.0)

if circuit\_type == "Integrator":

st.subheader("Integrator Simulation")

st.write("An op-amp integrator produces an output voltage that is proportional to the time integral of the input voltage.")

time\_constant = R \* C

st.write(f"Time Constant ($\tau$): {time\_constant:.4f} s")

st.markdown("[Image of an op-amp integrator circuit]")

time\_points = [i / 1000 for i in range(200)]

input\_signal = [vin\_amplitude \* math.copysign(1, math.sin(2 \* math.pi \* frequency \* t)) for t in time\_points]

output\_signal = []

current\_val = 0

for t in time\_points:

current\_val += -input\_signal[time\_points.index(t)] / (R \* C) \* (time\_points[1] - time\_points[0])

output\_signal.append(max(min(current\_val, vin\_amplitude \* 2), -vin\_amplitude \* 2))

else: # Differentiator

st.subheader("Differentiator Simulation")

st.write("An op-amp differentiator produces an output voltage that is proportional to the time derivative of the input voltage.")

time\_constant = R \* C

st.write(f"Time Constant ($\tau$): {time\_constant:.4f} s")

st.markdown("")

time\_points = [i / 1000 for i in range(200)]

input\_signal = [vin\_amplitude \* math.copysign(1, math.sin(2 \* math.pi \* frequency \* t)) for t in time\_points]

output\_signal = []

for i in range(len(input\_signal)):

if i > 0:

if input\_signal[i] != input\_signal[i-1]:

output\_signal.append(input\_signal[i] \* 10)

else:

output\_signal.append(0)

else:

output\_signal.append(0)

chart\_data = {

'Time (s)': time\_points,

'Input Voltage (V)': input\_signal,

'Output Voltage (V)': output\_signal

}

st.line\_chart(chart\_data, x='Time (s)', y=['Input Voltage (V)', 'Output Voltage (V)'])

# --- Main App Logic ---

if st.session\_state.page == 'home':

show\_home\_page()

elif st.session\_state.page == 'Basic Op-Amp Simulator':

show\_basic\_op\_amp\_simulator()

elif st.session\_state.page == 'Integrator/Differentiator':

show\_integrator\_differentiator()

else:

# Fallback page for experiments that are not yet implemented

st.header(st.session\_state.page)

st.button("Return to Home", on\_click=navigate\_to, args=('home',))

st.warning("This experiment is not yet implemented. Please select another one.")