****

**EE 586 Signal and Power Integrity**

**Project 2 Report**

Fall 2024

Submitted by:

**Dagmawi Abera**

**1. HFSS Design**

The first image shows the **ANSYS HFSS (High-Frequency Structure Simulator)** design of two closely spaced microstrip lines, used for analyzing signal propagation and **crosstalk** (NEXT and FEXT).

**Key Design Components:**

* **Substrate**: The green base material acts as the dielectric medium for the microstrip lines, typically FR4.
* **Two Microstrip Lines**:
  + **Orange Line**: Represents one of the signal traces.
  + **Blue Line**: Represents the other signal trace placed close to the first line to introduce crosstalk effects.
* **Ports**: The ports are placed at the start and end of each line:
  + **Port 1 and Port 3**: Input ports for the signal injection.
  + **Port 2 and Port 4**: Output ports for analyzing the signal's behavior, including S-parameters.

The objective of this design is to analyze the crosstalk between the two lines using **S-parameters**, where:

* S11​ and S22​: Input reflection coefficients.
* S21​ and S12​: Transmission coefficients, representing **Near-End Crosstalk (NEXT)** and **Far-End Crosstalk (FEXT)**.
* A screenshot of a computer

  Description automatically generated

**A computer screen shot of a computer program

Description automatically generated**

**2. S-Parameter Results**

The second image shows the **S-parameter results** obtained from the HFSS simulation.

**Explanation of S-Parameters:**

* **dB(S(1,1))**: Reflects the input signal loss at Port 1.
* **dB(S(2,1))**: Represents **Near-End Crosstalk (NEXT)**—how much signal at Port 1 couples into Port 2.
* **dB(S(1,2))**: Represents **Far-End Crosstalk (FEXT)**—how much signal at Port 2 couples into Port 1.
* **dB(S(2,2))**: Reflects the input signal loss at Port 2.

**Observations:**

* The S-parameter magnitudes (in dB) exhibit periodic variations due to signal reflections and interactions.
* **NEXT (S21)** and **FEXT (S12)** show significant coupling effects, especially at higher frequencies (e.g., around 15–20 GHz).
* S11​ and S22​ remain relatively low, indicating good signal retention at the input ports.

The extracted S21​ and S12​ values are later used in the Python simulation to simulate **NEXT** and **FEXT** signals.

A screen shot of a graph

Description automatically generated

**3. Python Code Explanation**

The Python code simulates **Near-End Crosstalk (NEXT)** and **Far-End Crosstalk (FEXT)** using the S-parameters extracted from HFSS.

**Steps in the Code:**

1. **CSV File Reading and Cleaning**:
   * The program loads the CSV file containing S-parameters (dB(S(2,1)), dB(S(1,2)), etc.).
   * Column names are cleaned to remove extra spaces and special characters for ease of access.
2. **S-Parameter Conversion**:
   * S21​ (NEXT) and S12​ (FEXT) are provided in **dB scale**.
   * The values are converted to **linear scale** using the formula:  
      Linear = 10^dB/20
3. **Time Domain Simulation**:
   * A **unit step function** is defined as the input signal.
   * **Interpolation** of S21​ and S12​ values ensures alignment with the simulation time range.
4. **Crosstalk Functions**:
   * **NEXT Simulation**:
     + NEXT is proportional to the derivative of the input signal.
     + The Python function multiplies the input signal with the interpolated S21S\_{21}S21​ coefficient.
   * **FEXT Simulation**:
     + FEXT is a delayed version of the input signal scaled by S12​.
     + A time delay is introduced using np.roll().
5. **Visualization**:
   * Plots the **Input Signal**, **NEXT**, and **FEXT** to show the signal behaviors clearly.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file

file\_path = r"C:\Users\dagit\OneDrive\Desktop\Fall 2024\EE 486,486 Signal and Power Integrity\HFSS Project\Project 2\S Parameter Plot 1.csv"

try:

data = pd.read\_csv(file\_path)

except FileNotFoundError:

print(f"Error: File not found at '{file\_path}'. Check the path and try again.")

exit()

# Clean column names by removing spaces, brackets, and special characters

data.columns = data.columns.str.strip().str.replace(r"[^\w]", "", regex=True)

# Debugging: Print cleaned column names

print("Cleaned Column Names:", data.columns)

# Extract relevant columns

try:

frequency = data['FreqGHz'] \* 1e9 # Convert GHz to Hz

S21\_dB = data['dBS21'] # NEXT data in dB

S12\_dB = data['dBS12'] # FEXT data in dB

except KeyError as e:

print(f"Error: {e}. Check your file for column names.")

exit()

# Convert S-parameters from dB to linear scale

S21\_linear = 10 \*\* (S21\_dB / 20) # NEXT coefficient

S12\_linear = 10 \*\* (S12\_dB / 20) # FEXT coefficient

# Time settings for simulation

time = np.linspace(0, 2e-9, 1000) # Time range (0 to 2 ns)

unit\_step = np.heaviside(time, 1) # Unit step input signal

# Interpolate S-parameter coefficients for the time domain

NEXT\_coeff = np.interp(time, np.linspace(0, max(time), len(S21\_linear)), S21\_linear)

FEXT\_coeff = np.interp(time, np.linspace(0, max(time), len(S12\_linear)), S12\_linear)

# Crosstalk transfer functions

def NEXT\_signal(input\_signal, coeff):

"""Simulate Near-End Crosstalk (NEXT)."""

next\_signal = coeff \* np.diff(input\_signal, prepend=0)

return next\_signal

def FEXT\_signal(input\_signal, coeff, delay=50):

"""Simulate Far-End Crosstalk (FEXT)."""

delayed\_signal = np.roll(input\_signal, delay)

delayed\_signal[:delay] = 0 # Zero out initial delay

fext\_signal = coeff \* delayed\_signal

return fext\_signal

# Simulate the signals

next\_output = NEXT\_signal(unit\_step, NEXT\_coeff)

fext\_output = FEXT\_signal(unit\_step, FEXT\_coeff, delay=50)

# Plot results

plt.figure(figsize=(10, 8))

# Input Signal

plt.subplot(3, 1, 1)

plt.plot(time, unit\_step, color='blue')

plt.title("Input Signal (Unit Step)")

plt.xlabel("Time (s)")

plt.ylabel("Amplitude")

plt.grid()

# Near-End Crosstalk (NEXT)

plt.subplot(3, 1, 2)

plt.plot(time, next\_output, color='red')

plt.title("Near-End Crosstalk (NEXT)")

plt.xlabel("Time (s)")

plt.ylabel("Amplitude")

plt.grid()

# Far-End Crosstalk (FEXT)

plt.subplot(3, 1, 3)

plt.plot(time, fext\_output, color='green')

plt.title("Far-End Crosstalk (FEXT)")

plt.xlabel("Time (s)")

plt.ylabel("Amplitude")

plt.grid()

plt.tight\_layout()

plt.show()

**4. Simulated Results**

The third image displays the simulation results for the **input signal**, **NEXT**, and **FEXT**.

**Plot Breakdown:**

1. **Input Signal (Top Plot)**:
   * The input is a **unit step function**, which jumps from 0 to 1 at time t=0.
   * This represents the driving signal injected into the microstrip line.
2. **Near-End Crosstalk (NEXT) (Middle Plot)**:
   * The NEXT signal shows a sharp pulse at t=0, which corresponds to the derivative of the unit step signal.
   * The amplitude of the NEXT signal is scaled by the S21​ coefficient (NEXT coupling strength).
3. **Far-End Crosstalk (FEXT) (Bottom Plot)**:
   * The FEXT signal shows a delayed and weaker version of the input signal.
   * The delay occurs because FEXT happens at the far end of the line and takes time to propagate.
   * The signal's amplitude is scaled by the S12​ coefficient (FEXT coupling strength).

A screenshot of a computer screen

Description automatically generated

**5. Conclusion**

1. **HFSS Design**:
   * The HFSS model successfully simulates the microstrip line configuration and calculates the S-parameters.
2. **S-Parameter Results**:
   * S21​ (NEXT) and S12​ (FEXT) demonstrate the coupling effects between two closely spaced lines.
3. **Python Simulation**:
   * The Python code utilizes the extracted S-parameters to model **NEXT** and **FEXT** in the time domain for a unit step input signal.
   * The results validate the crosstalk behavior, showing both the **derivative nature** of NEXT and the **delayed nature** of FEXT.
4. **Visualization**:
   * The final plot provides a clear representation of the input signal and its coupled effects (NEXT and FEXT).