Project: Netlist Capacitance Calculation using TCL

Introduction:

In real world, we use tcl script along with EDA tools to maximize flexibility and empower tools ability to perform given tasks. One of the common tasks is to use tcl to make calculations. In this project, we will use our knowledge in tcl scripting to perform such a calculation. Tcl power expands beyond this tasks as well.

Objective:

To develop a TCL script that reads netlist file and a cell library file, identifies specific net paths as specified by the user, and calculates the total capacitance for each path based on the capacitance values from the cell library. This project reflects real-world applications of TCL in digital design and encourages you to think about appropriate data structure, appropriate algorithm and file manipulation.

We will divide this project into subprojects to make it easier to follow.

Subproject#1: Capacitance extraction from Lib file

Given a standard cell library, we want to extract the max capacitance of each gate.

Example on standard cell library:

```
/*******************************************************************************************
  Module                : INV_X1
  Cell Description      : Combinational cell (INV_X1) with drive strength X1
 *******************************************************************************************/
cell (INV_X1) {

        drive_strength          : 1;
        area            : 0.532000;
        pg_pin(VDD) {
                voltage_name : VDD;
                pg_type      : primary_power;
        }
        …….
        pin (ZN) {

                direction               : output;
                related_power_pin       : "VDD";
                related_ground_pin      : "VSS";
                max_capacitance                 : 60.730000;
                ……
```

Requirements:

Your task is to read such a library file, extract each cell name and the corresponding max_capacitance in a dictionary or associative array.

Example output:

```
{" INV_X1" 60.730000  "AND2_X1" 0.5 "INV_X1" 0.3 "NAND2_X1" 0.4 }
```

Hint:

- Think how to get a value from within the matched pattern,
- The order you get cellnames is the same order you can get the capacitance
- Take care that in the lib file cell (INV_X1) is not unique as it is mentioned in the comments, make sure to make a regex to get the cell name once only.

Subproject#2: Create Utilities to calculate capacitance

This subproject is quite simple but tricky. Assume we already have two capacitance, we need to get the equivalence of such capacitance. Equivalent capacitance is calculated as follows:

The equivalent capacitance of capacitors connected in **parallel** is the sum of their individual capacitances. When capacitors are connected in parallel, they have the same voltage across them, and their charges add up. For two capacitors C_1 and C_2 connected in **parallel**, the **equivalent capacitance** C_{eq} is given by:

$$C_{eq} = C_1 + C_2$$

The equivalent capacitance of capacitors connected in **series** is found using the reciprocal of the sum of the reciprocals of their individual capacitances. When capacitors are in series, they carry the same charge, but the total voltage across the series combination is the sum of the voltages across each capacitor. For two capacitors C_1 and C_2 in series, the **equivalent capacitance** C_{eq} is given by:

$$\frac{1}{C_{eq}} = \frac{1}{C_1} + \frac{1}{C_2}$$

Requirements:

Create a tcl proc “calculate_capacitance” that takes array of capacitance values and a flag to state whether they are parallel or series. The proc calculates the capacitance and return the equivalence based C_{eq} on the equations above.

Example:

```
Calculate_capacitance { 1 2 3 4}  1
#where 1 stands for parallel, this should result in 10
Calculate_capacitance { 1 2 3 4}  0
#where 0 stands for series, this should result in 0.48
```

Subproject#3: Extract path from Netlist

In this subproject, we want to extract paths from a netlist and get the gates a long the path. This task requires a good algorithm and appropriate data structure to make it possible. You can start by the input or by the output, whatever you choose.

For Example, for the given code

```
module datapath(inputB, inputA, p_0);
   input [7:0]inputB;
   input [7:0]inputA;
   output [8:0]p_0;

   HA_X1 i_0 (.A(inputB[0]), .B(inputA[0]), .CO(n_0), .S(p_0[0]));
   HA_X1 i_1 (.A(inputB[1]), .B(inputA[1]), .CO(n_2), .S(n_1));
   HA_X1 i_2 (.A(inputB[2]), .B(inputA[2]), .CO(n_6), .S(n_5));
   HA_X1 i_3 (.A(inputB[3]), .B(inputA[3]), .CO(n_10), .S(n_9));
   XOR2_X1 i_8 (.A(n_0), .B(n_1), .Z(p_0[1]));
  //……
```

- ➔ Extract Path inputB[0]:p_0[0]
  - Answer is: {S, HA_X1 (i_0) }
  - S stands that the following list is connected in series
- ➔ Extract Path inputB[0]:p_0[1]
  - Answer is: {S, HA_X1 (i_0), XOR2_X1(i_8) }

Parallel Circuit Example:

```
module datapath(A, B, C, K );
   input A, B , C;
   output K,L,M;
   AND2_X1 i_0 (.A(A),.B(B),.ZN(n_0));
   OR2_X1 i_1 (.A(A),.B(C),.ZN(n_1));
   AND2_X1 i_2 (.A(n_0),.B(n_1),.ZN(n_2));
    INV_X1 i_3 (.A(n_2), .ZN(K));
endmodule
```

- ➔ Extract Path A:K
  - Answer is: {S, { P, AND2_X1 (i_0), OR2_X1(i_1) }, AND2_X1 (i_2) , INV_X1 (i_3)}
  - P stands that the following list are connected in parallel.

Hint:

- ➔ You can build adjacency list for each input/intermediate output to know how to trace the path
- ➔ Take care of parallel paths this would be a bit tricky to identify
- ➔ **Optional** (get a path from nested modules)

Subproject#4: Integration

Given subprojects #1,#2,#3, you are now ready to calculate the total resistance on a given path.

Project Requirements

1. **Input Files:**
   - **Netlist**: Each netlist is a text file containing instances and their interconnections.
   - **Cell Library**: A file that lists each cell type and its capacitance. Each cell in the library has a unique identifier (e.g., "AND2", "OR2", etc.) and a corresponding capacitance value.
2. **User Input:**
   - The user will specify the names of net paths they wish to analyze. A path is defined using two points, the beginning point (input) and the end point(output).
     - Example: Path A:B is a path that starts at A and ends at B
   - Top modules: user will specify your top module, where you will start parsing.
   - User inputs should be command line parameters as follows:
   - ```
     >project.tcl -path "A:B"  -top: "Adder"
     ```
3. **Output:**
   - the script will output the total capacitance.

Hint:

- ➔ Extract capacitance from subproject#1 and store it in some Data structure
- ➔ Extract path from netlist using subproject#3.
- ➔ Using the capacitance from subproject#1, replace every element in the path with the equivalence capacitance.
- ➔ Use subproject#2 to calculate total capacitance
     ```