C:\Users\Mostafa\Desktop\nameplate.jpgwhaxtent AUC stud

**MiniProject**

**Path Delay Calculator Utility**

**(Python)**

**To**

Dr. Mohamed Shalan

**By**

Khaled Soliman ID: 900162031

Mohamed Samir ID: 900150936

Due date: 12-11-2019

**Introduction**

A utility that calculates the delay of a path taking into consideration the interconnect effects. The utility accepts a routed DEF file, the technology LEF and the library liberty file. The user has to specify the path cells in a file.

**Delay Calculation Approach**

**Stage 1: Input Validation**

Before starting to calculate the delay of a given path, it is first necessary to validate the connectivity of this path otherwise the calculations which will take place afterward would be useless. Hence, we first read the input file from the user in which components and pins re defined. Then we make sure that each of those component/pin precede to the next component/pin, iterating on the entire input file till reaching the last defined component/pin.

**Stage 2: Cell Delay**

After the validation stage we then start calculating the delay of the cells. Though, first we have to get the capacitive load of each net which we get by getting the capacitance of each cell that is in the net we are dealing with from the slandered cell library and add them all up. Finally, taking both values of the transition time which is 0 when first starting and look for the delay for these two values inside the transition tables existed in the liberty file. This is just a general idea about how cell delay calculation was approach, more into that is documented in details within the submitted code.

**Stage 3: Interconnect Delay**

To get the interconnect delay we have first to get the wire length and width which we get by subtracting the x-axes and the y-axis coordinates defined in DEF file. Then to be able to actually get the resistance and the capacitance and the multiply them to have the delay across each wire. We use both the C-Sheet and the R-Sheet tables existed in the and the R-Sheet defined in the LEF file. Finally applying Elmore Delay Model tpd = R1C1 + (R1+R2)C2 + (R1+R2+R3)C3 + …. To get the total delay.

**Test Cases**

There are 11 test cases that are provided in the Example folder submitted. First 10 cases are actually valid, meaning that they consist of cells/pins that have interconnection with one another. However, test case number 11 is a faulty one, meaning it consists of cells/pin that have no interconnection with one another. Hence, it is made sure that our project functions in both valid and fault cases.

**Bonus Part**

In the bonus part what is added is that instead asking the user to provide the path cells one by one of providing the path cells the user is onl*y required to* provide the path starting and ending points. This of course had us to go figure out all the possible paths between the two terminal points and then check which of those paths actually connects to each other and once found, we insert them in a linked list. Afterwards we just return this linked list to our algorithm to run on it once more but this time with the cells given and then it will return all possible delay in an array which we will then output the maximum one out of them.

**Bonus Part Test Cases**

The test cases for the bonus part is basically the same as the test cases provided, yet it is edited to only have the first and the last component/pin only and then the algorithm will figure out the path and calculate the delay.

**Github Link:**

<https://github.com/KhaledSoliman/MiniProjectDigital>

**References**

Liberty User Guides and Reference Manual Suite Version 2013.03

LEF/DEF Language Reference Product Version 5.7 November 2009

Delay\_Calculation.pdf

Standard\_Cell\_Library.pdf

https://pypi.org/project/liberty-parser/

https://github.com/trimcao/lef-parser