Cell Propagation Delay calculation

Overview:

Propagation delay is an important parameter in digital circuit design, and it represents the time it takes for a signal to travel from the input of a logic gate to its output. In .lib (library) files, which are used in electronic design automation (EDA) tools, propagation delay information is crucial for accurate simulation and analysis of digital circuits.

The assignment's aim is to create a script that can easily figure out the propagation delay of a cell. All it needs are basic details like the cell name, transition time, output capacitance, and whether it's rising or falling. This is handy because regular libraries are often complex, and doing these calculations by hand can be tough and time-consuming. The script is intended to simplify and speed up the process of finding the propagation delay.

.lib file (Liberty file)

A .lib file is an ASCII representation of the Timing, Power and Area associated with the standard cells. It is a critical component in electronic design automation (EDA) tools for digital circuit design. It contains information about the timing, power, and other characteristics of various standard cells or library elements used in digital integrated circuit design. Here's an abstract of what a .lib file typically includes:

- **Library information:** contains wire load model and operating conditions.
- **Cell Definitions:** contains cell Name, cell function.
- **Timing Information:** contains timing models, T_{hl}, T_{lh}, T_{cq} and transition time.
- **Power Characteristics:** contains Static, Dynamic and leakage power.
- **Pin Information**: contains pin name, direction and capacitance.

In .lib file we use NLDM to model timing characteristics of cells like propagation delay and transition time,

Script's Algorithm:

I have developed a TCL script to compute the propagation delay, employing a series of steps. The initial phase involves parsing pertinent data from a .lib file, encompassing cell information, Look-Up Tables (LUTs), and timing delays. Subsequently, the script undergoes a processing stage where it takes input transition and output capacitance, entering these values into the LUT to determine the Cell Rise Delay or Fall Rise Delay and subsequently calculates their average. The final stage involves the script generating output files, preserving information related to the cell and LUT, and storing the computed propagation delay.:

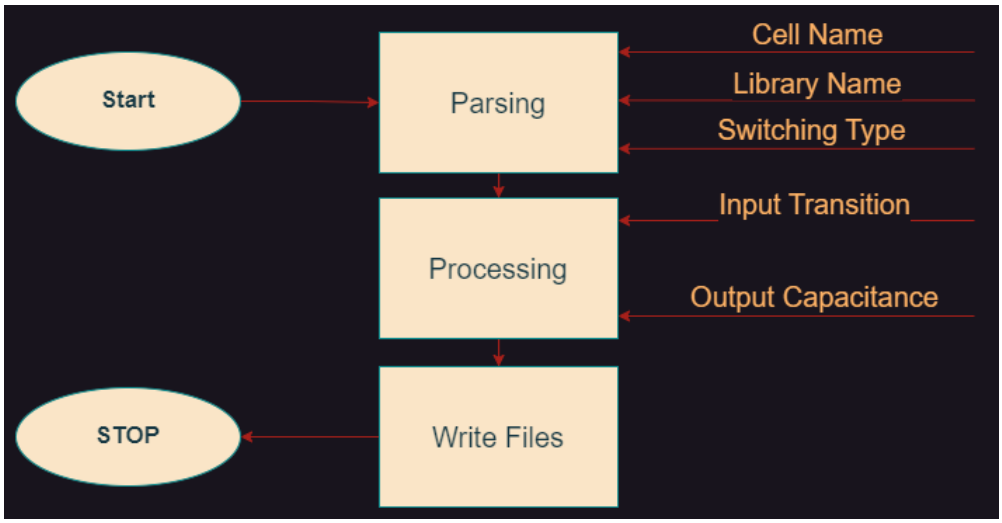

Figure 1 Script Steps

1. Parsing:

In the parsing step, the script utilizes regular expressions to systematically extract specific data from a sizable .lib file. Regular expressions help precisely identify patterns and structures within the file, making it efficient to retrieve essential information. For example, when parsing Look-Up Tables (LUTs), the script employs regular expressions to pinpoint and capture the LUT data, which is then stored in variables. These variables act as organized repositories of crucial information, ready to seamlessly integrate into the subsequent processing step. This approach ensures a focused and effective extraction of relevant details, as illustrated in the specific case of parsing and storing LUT data from the .lib file.

```
lu_table_template(x1_130_6x10) {
variable_1 : input_net_transition ;
variable_2 : total_output_net_capacitance ;
index_1( "  20.0,   60.0,   90.0,  130.0,  200.0,  300.0,  450.0,  670.0, 1000.0, 1500.0" ) ;
index_2( "   1.0,    4.0,   12.0,   30.0,   62.0,  130.0" ) ; }
```

Figure 2 Parsing Example

2. processing:

In the processing step, the script examines input transition and output capacitance values by checking their existence in the Look-Up Tables (LUTs). If both values are found in the LUTs, the script proceeds to the cell timing information to retrieve the matched value. However, if either or both values are not present in the LUTs, the script resorts to mathematical calculations. It performs interpolation when the values are within the LUT range and extrapolation when they fall outside the defined range. It's worth noting that while interpolation provides higher accuracy, extrapolation is employed when necessary, acknowledging its inherent lower accuracy compared to interpolation.

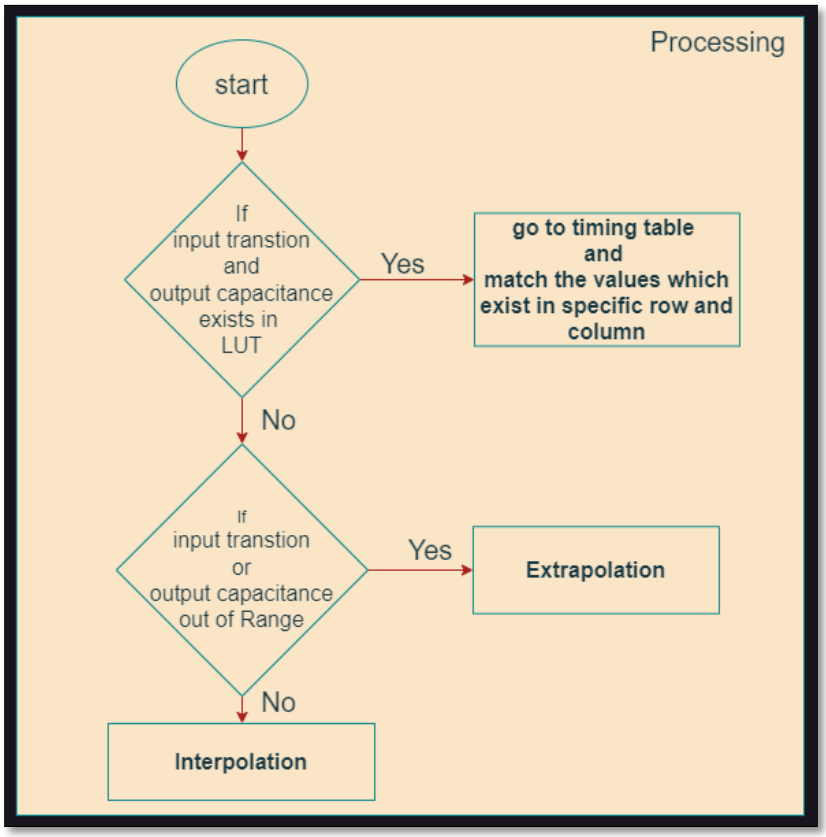

Figure 3 Processing Step

Computing interpolation and extrapolation:

The upcoming presentation will feature equations crucial for interpolation and extrapolation, integral to the calculations within the script. These mathematical expressions are sourced from a paper detailing various delay models, and a link to the figure will be provided for reference.

If the table lookup is required for (x_0, y_0), the lookup value T_{00} is obtained by interpolation and is given by:

$$T_{00} = x_{20} * y_{20} * T_{11} + x_{20} * y_{01} * T_{12} + x_{01} * y_{20} * T_{21} + x_{01} * y_{01} * T_{22}$$

Where:

$$x_{01} = (x_0 - x_1) / (x_2 - x_1)$$
$$x_{20} = (x_2 - x_0) / (x_2 - x_1)$$
$$y_{01} = (y_0 - y_1) / (y_2 - y_1)$$
$$y_{20} = (y_2 - y_0) / (y_2 - y_1)$$

Substituting 0.15 for *index_1* and 1.16 for *index_2* results in the fall_transition value of:

$$T_{00} = 0.75 * 0.25 * 0.1937 + 0.75 * 0.75 * 0.7280 + 0.25 * 0.25 * 0.2327 + 0.25 * 0.72 * 0.7676 = 0.6043$$

Note that the equations above are valid for interpolation as well as extrapolation – that is when the indices (x_0, y_0) lie outside the characterized range of indices. As an example, for the table lookup with 0.05 for *index_1* and 1.7 for *index_2*, the fall transition value is obtained as:

$$T_{00} = 1.25 * (-0.25) * 0.1937 + 1.25 * 1.25 * 0.7280 + (-0.25) * (-0.25) * 0.2327 + (-0.25) * 1.25 * 0.7676 = 0.8516$$

Figure 4 Interpolation and Extrapolation Equations

3. Generating Output Files:

In the final step of generating output files, the script captures parsed information for ease of interpretation. It initiates the storage of key parsing operations, such as cell information and Look-Up Tables (LUTs), in dedicated output files. This practice facilitates easy retrieval and review of essential details. Furthermore, an abstract output is created for each tested cell, encompassing crucial parameters such as cell name, input transition, output capacitance, type of switching, and the calculated propagation delay.

Script validation

I have successfully validated my script by applying various test cases and comparing the results with manual calculations. It's noteworthy that the manual tests were conducted specifically on matched cases, excluding interpolation or extrapolation scenarios. The table below provides details on some of the test cases:

Test Case	Cell Name	Input Transition	Output Capacitance	Switching Type	Manual Calculation	Script Result
1	no4_x1	20.0	1.0	Rising	89.15	89.15
2	nao2o22_x4	90.0	124.8	Rising	288.1	288.1
3	nao2o22_x4	90.0	124.8	Falling	247.9625	247.9625
4	nao2o22_x4	1600	124.8	Rising	-	484.4681
5	nmx2_x1	200.0	2.6	Falling	64.7	64.7
6	nmx2_x1	200.1	2.0	Falling	-	49.6682