**CSCE 337 - Digital Design II**

**Dr. Mohamed Shalan**

**GTA: Marwan Adel**

**Project 2: Static-Timing-Analysis**

**By:**

**Yehia Hesham**

**Ahmad Abdalla**

**Tasneem Ismail**

**Overall Progress:**

1. We used the Gate-Level Netlist parser and the DAG structure from the previous project, after doing minor modifications.
2. We were able to parse all files needed for the timing analysis. That includes: Net Capacitance file, Clock Skew file, Initial Input transition file, Clock period and input delay file.
3. We parsed the Liberty file and were able to extract all needed values from the tables (Gate delay, output transition and pin capacitance) using Interpolation and Extrapolation.
4. We were able to extract full Paths from every input, passing by the respective pins, to every output, and identified all the needed timing paths (REG-REG & IN-REG).
5. We were able to get the output capacitance on the output of every gate.
6. We were able to get the input transition of every input pin of every gate.
7. We wrote a Sudocode for sorting the paths based on their slacks.
8. We created 6 test cases with different complexities. But, we only had time to do analysis by hand for 2 of them.

**Problems:**

1. We admittedly wasted time trying to integrate the liberty parser with the other data structures and class.
2. We wasted even more time getting around the code and how to use the needed functions (particularly the Cell\_Delay() and the Cell\_Set\_Hold() functions).
3. This has made us late on schedule, and therefore unable to begin the timing analysis phase, which could have been done in 2 days maximum, especially since we are in the finals week.

**Parsers Design:**

* **The main class is parser.h, it consists of some functions:**
* vector<Cell> Read(string lib\_name); //reads liberity file and done all parsing and return a vector of different Cells (class will be explained later)
* int char\_int(char c); //converts from char to int, is needed in parsing
* float Cell\_Delay(string cell, string pin, float index1, float index2, int b); //it returns cell delay, given five different parameters, string cell: is the type of cell, ie NAND2X1. String pin: is the name of pin ie A, or B, and if for FF it would be clkh/clks, index1 is output capacitance, index2 is input transition, b is either zero or 1, 1 means rising and 0 means falling //private function and its equivalent user interface is float getCellDelay(string cell, string pin, float outputcap, float inputtrans);
* float Cell\_Transition(string cell, string pin, float index1, float index2, int b); //same as previous function //private function and its equivalent user interface is float getCellTransition(string cell, string pin, float outputcap, float inputtrans);
* float Cell\_Set\_Hold(string cell, string pin, float index1, float index2, int b); //index1: related\_pin\_transition, ie CLKh, CLKs. Index2: constrained\_pin\_transition, ie D= date of flipflop
* float input\_capac(string cell, string pin, int b); //b if 0->> returns fall capacitance, b if 1 ->> returns rise capacitance, if anything else->> returns total capacitance
* Cell Return\_Cell(string name); //used to return a cell given name, ie NAND2X1 //private function
* PinI Return\_Pin(string cell,string name); //used to return a Pin given its name ie, A or B, given cell name, ie NAND2X1 //private function
* **We have class PinO**

// it represents the output pin and its name and capacitances.

* **We have class PinI**

//it represents input pin and its name, capacitances, vectors for delays, transition,…etc

* **We have class Cell**

// it represents Cell, with its name, related input pins, output pin.

* **We have class StaticTiAnalysis**
* StaticTiAnalysis();
* ~StaticTiAnalysis();
* //NELIST METHODS
* void ReadNetlist(string filelocation);
* void writeJsonfile(string filelocation);
* void CreateGraphy();
* //liberatiy Method
* void ReadLib(string loc);
* //DAG,PATHS METHODS
* void printPaths();
* void topologicalSortUtil(int s, int v, vector<cell> &adj, stack<string> &Stack, bool tillReg, bool& WE, bool& reg\_flag, double input\_net\_trans); //part with makes the DFS
* void addEdge(string v, string t, string w); //to add a directed edge
* int get\_index(string v, vector<cell> adj); //to get a specific index of a node since there is no find dunction in vectors
* void topologicalSort(); //this function loops on all the nodes and do DFS (using topological logic)to get the correct level
* //SKEW ,Delat, & Capacitance METHODS
* void StaticTiAnalysis::ReadSkewFile(string filelocation);
* float StaticTiAnalysis::GetSkew(string ffname);
* void StaticTiAnalysis::ReadWireCapFile(string filelocation);
* float StaticTiAnalysis::GetWireCap(string wirename);
* void StaticTiAnalysis::ReadConstraints(string filelocation);
* int StaticTiAnalysis::GetInputDelay(string input);
* //variable members
* vector<FFSkew> SkewsArray;
* int clockperiod;
* vector<InDelay> inputdelay;
* vector<wire> WireCap;
* vector<module> nodes;
* vector<string>p; //for paths
* vector<int>wires;
* int highest=0;
* Parser extract;
* vector<cell> adj; //this is a vector of a struct which will carry each node's info : outwards edges , name, type, etc
* **TestCode:**

StaticTiAnalysis test;

string f1 = "D:\\Auc Courses\\Spring 2015\\Digital Design II\\Projects\\Static Timing Analysis\\Static Tiiming\\Beautifier\\TestCase1\_yehia\\netlistfile.txt";

string f2 = "D:\\Auc Courses\\Spring 2015\\Digital Design II\\Projects\\Static Timing Analysis\\Static Tiiming\\Beautifier\\TestCase1\_yehia\\jsonfile.txt";

string f3 = "D:\\Auc Courses\\Spring 2015\\Digital Design II\\Projects\\Static Timing Analysis\\Static Tiiming\\Beautifier\\TestCase1\_yehia\\my\_skewfile.skw";

string f4 = "D:\\Auc Courses\\Spring 2015\\Digital Design II\\Projects\\Static Timing Analysis\\Static Tiiming\\Beautifier\\TestCase1\_yehia\\my\_inputConstraints.txt";

string f5 = "D:\\Auc Courses\\Spring 2015\\Digital Design II\\Projects\\Static Timing Analysis\\Static Tiiming\\Beautifier\\TestCase1\_yehia\\my\_netcap.ncp";

string f6 = "D:\\Auc Courses\\Spring 2015\\Digital Design II\\Projects\\Static Timing Analysis\\Static Tiiming\\Beautifier\\TestCase1\_yehia\\Liberity.lib";

test.ReadLib(f6);

test.ReadNetlist(f1);

test.ReadSkewFile(f3);

test.ReadConstraints(f4);

test.ReadWireCapFile(f5);

test.CreateGraphy();

test.writeJsonfile(f2);

test.printPaths();