LOGIC CIRCUITS BEAUTIFIER

Digital Design II project

Omar Khafagi
Eman H.Ahmed
Abdel Rahman EL Dallal

Outline

1- Introduction

In this project, we are required to design a gate level beautifier that reads a Verilog gate level netlist then converts it to a DAG (Directed Acyclic Graph). Finally apply a topological sorting algorithm on the constructed DAG. The outcome of the topological sort should be used to identify the logic level of each gate.

2- Design and implementation

The programming languages used for this project are matlab and perl. The reason for using these languages is that matlab toolboxes has functions for topological sorting and for creating the DAG (Directed Acyclic Graph). We also used perl libraries in the parsing.

Created Matlab Functions:

digitalproject1.m:

The maestro function that returns the JSON Data of the logical level of all of the nodes and Topological Sorting of the nodes, as well as a drawn Directed Acyclic Graph, topologically sorted, of the Verilog signals and modules. This function only calls parseVtn and uses the two vectors to create the Directed Acyclic Graph (the source node and destination node vectors), and draws and extrapolates the information from the graph to convert it into JSON data using the open-source JSONLab library. The function's only argument is the name of the Verilog file, disregarding the .g.v extension. For example, to parse booth.g.v, the argument to this function would simply be 'booth'.

insertarr.m:

A function that takes as parameters an array, an array element, and a numerical index. The function essentially inserts the array element into the array the specified index, shifting everything existing after that index in the array by one index value.

insertcell.m:

The same exact function as insertarr, but only for cell arrays instead of regular arrays.

parseVtn.m:

The function that does all of the parsing. It takes as arguments the name of the Verilog file, which is in turn passed as arguments to the new.pl, newtypes.pl, and newmod.pl, that return the names of the signals, their types, and the names of the modules, respectively. Using the lib.mat and libio.mat files (generated by the libref.pl script) alongside the aforementioned scripts, the function relates the signals as either outputs or inputs to the module, and returns the input vector and output vector (representing the source and destination nodes respectively) for the main function to use.

sortother.m:

Takes two arrays as an argument, and performs bubble sort on the second array based on the elements of the first.

Created Perl Scripts:

libref.pl

Parses the osu035_stdcells.v file and creates a comma separated output containing the names of each standard Verilog module, followed by a space, followed by a sequence of dash separated 0s and 1s that indicate module parameters as either outputs (0s), or inputs (1s), in the correct parameter passing order. This script was used to create the lib.mat and libio.mat files, which are importable files created by Matlab, which represent two Matlab cells containing strings; lib.mat containing the names of the modules, and libio.mat containing the dash separated *binary* sequence. These two cell arrays are simply a reference for the Verilog parser Matlab function to parse to understand what a Verilog module looks like, and which argument represents the input and output signals.

new.pl

Takes as argument the Verilog file (or files) to parse, and for each module, returns its signal names, using the rvp.pm Perl library.

newtypes.pl

Takes the same argument as new.pl, and returns the signal types in the same order as new.pl.

newmod.pl

Takes the same argument as new.pl, and returns all Verilog code found in the modules, while ignoring comments, declarations. Assignment statements and other irrelevant information are ignored by the Verilog parser itself.

Libraries Used

rvp.pm

A Perl library that contains many useful build in functions to parse Verilog modules. It has the ability to identify declared modules (but not pre-existing modules), signals, Verilog comments and declarations. All previously mentioned Perl scripts use this library to achieve their functionality.

JSONLab

A Matlab library only used to convert the information stored as Cell Arrays into JSON Arrays. It contains many other useful functions such as parsing JSON data, but they are not used for the purpose of this project.

3- Testing Strategy

For the test cases, we designed test cases that tests how efficient the results are. The test cases purpose was to test different gate level netlists. Also, we tested it for huge designs to see whether it will work or not. We tested the code for all the samples that were provided on the black board and the samples we did.