Logic Simulator

v0.85

Sai Bhargav Yalamanchi
Shaimak Reddy

Indian Institute of Technology, Bombay

Contents

1. INTRODUCTION

The logic simulator takes in any well defined circuit and generates the logic-time waveforms at every node of the circuit.

Delays can be specified for every circuit component.

2. INSTALLATION

At the time of writing this document, only Linux based operating systems supported the Logic Simulator.

The following are the requisite packages necessary for proper functioning of the program:

1. Icarus Verilog

http://iverilog.wikia.com/wiki/Installation_Guide

The Ubuntu Universe repository has the Icarus Verilog .deb package. To install Icarus Verilog follow the steps below

•Add the Universe repository in /etc/apt/sources.list using your favourite text editor (It would already be there but would have been commented). In Edgy Eft,the line would be something like
deb http://archive.ubuntu.com/ubuntu/ edgy universe
deb-src http://archive.ubuntu.com/ubuntu/ edgy universe
•Run **sudo apt-get update** from the terminal
•Run **sudo apt-get install verilog**

2. Python

Download the Python bzipped source tarball here -
http://www.python.org/ftp/python/2.7.3/Python-2.7.3.tgz

3. graph-tool

Instructions for installation can be found here -
http://projects.skewed.de/graph-tool/wiki/GraphToolDownload

All dependencies will be taken care of automatically.

3. THE INPUT

The circuit is represented by the GraphML File Format. An introduction to GraphML can be found here - http://graphml.graphdrawing.org/primer/graphml-primer.html

The circuit is represented by a graph. Each circuit component constitutes a node, and each connecting link an edge. Each node is packed with metadata, which is to be specified by the user.
Metadata for nodes -

1. num_inputs - The total number of **primary inputs**. Primary inputs are those inputs that are

directly controlled by the user. An output of a gate that drives another is not a primary input. It is zero by default.

2. type - The class of the node. Valid values are “and”, “or”, “not”, “nor”, “xor”.
3. pos - It is 1 only if the node has a **primary output**. Primary outputs are those outputs that do not drive any other gates. Such nodes can be thought of as terminal nodes. Default – 0.
4. delay - The propagation delay. Must be a whole number. Default – 0.

Metadata for the graph -

1. inputs - The total number of primary inputs
2. outputs - The total number of primary outputs

The metadata is supplied by means of a key-value pair system.
Property keys for such a system must be defined in the xml file.
The values are enclosed within a “data” tag, whose “key” attribute value must match the “id” attribute value of the corresponding property key.

For example, I wish to set 10 as the number of inputs (“inputs”).
I define the property key as -

```
<key id="key0" for="graph" attr.name="inputs" attr.type="int" />
```

And I write (after opening the graph tag) -

```
<data key="key0">10</data>
```

The **basic structure** of the xml file will resemble the following -

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">

  <!-- property keys -->

  <graph id="G" edgedefault="directed">

   <!-- graph properties -->

   <!-- vertices -->

   <!-- edges -->
  </graph>
</graphml>
```

Note that the “id” attribute value must be uniquely specified.

The vertices are defined thus -

```
  <node id="n0">
    <data key="key1">2</data>
    <data key="key3">0</data>
    <data key="key2">and</data>
```

```
        </node>
```

The edges are defined thus -

```
    <edge id="e0" source="n0" target="n2">
    </edge>
```

The graph is directed, so the source-target ordering is important.

4. HOW TO RUN THE PROGRAM

There are two ways to run the program -
1. Either enter the command

```
    python makecircuit.py <outfile> <infile>
```

where <outfile> is the file name of the output verilog file that will be generated by the script, and <infile> is the file name/address of the input graphml file.

2. If the bash script “run” is to be run for the first time, enter the command

```
    chmod +x run
```

Then enter

```
    ./run
```

5. THE TEST BENCH

The code generates a compilable testbench.

In the 'initial' block enter the relevant signal information.

Compile the file and view the output in any standard waveform viewer such as GTKWave.

To compile the file, enter the command

```
    iverilog <file> lib/*.v
```

where <file> is the name of the verilog file that the script generates.