V2Tau: Verilog to LSGS Format Conversion Tool

Version 0.2

Users' Guide

Kyle Kelley   Siddharth Joshi   Stephen Boyd

Electrical Engineering Department
Stanford University

`kkelley@stanford.edu`   `sidj@stanford.edu`   `boyd@stanford.edu`

May 28, 2007

The large-scale gate sizing (LSGS) MATLAB toolbox [JB07] accepts data for the gate sizing optimization problem in a particular format described in [JB06]. The conversion tool V2Tau translates a Verilog netlist into the particular format required by LSGS, and also back annotates the resulting gate sizes into the Verilog netlist. It consists of two Perl scripts: `v2tau.pl` and `tau2v.pl`. The Perl script `v2tau.pl` reads a Verilog netlist, and produces an m-file which, when executed in MATLAB, sets up the gate sizing optimization problem in the format required by LSGS. The LSGS toolbox can then solve the gate sizing problem. The Perl script `tau2v.pl` can then be used to back annotate the gate sizes found by LSGS into the original Verilog netlist. Thus the conversion tool V2Tau provides a front-end for the LSGS toolbox, so that it can be used within a typical physical circuit design flow.

Since all inputs to LSGS (except the area coefficients) have units of time, typically RC time constants denoted by τ, the conversion tool has been named V2Tau.

Setup. First, ensure that all required Perl libraries are installed on the host system. The conversion tool V2Tau requires the following two Perl packages: `Verilog::Netlist` and `Graph`. One way to install these packages under Linux is via the comprehensive Perl archive network (CPAN). To install the `Verilog::Netlist` package run the following command as superuser (root)

```
(root)% perl -MCPAN -e 'install Verilog::Netlist'
```

To set up V2Tau conversion tool, download and extract the files to the directory `$BASE = $DD/v2tau-<ver>`, where `$DD` the directory the package has been extracted in, and `<ver>`

is the version number (0.2 in this case). The `$BASE` directory should have two subdirectories `src` and `examples`, and this users' guide. Set the environment variables `PATH` and `PERL5LIB` to include the directory `$BASE/src`. This enables running the Perl scripts from any directory.

Usage. The Perl script `v2tau.pl` takes in three arguments. The first argument is a Verilog netlist of the circuit. The second argument contains the values of the wire load capacitances of the circuit. The third argument is a Perl module which contains parameter values for the delay model of the gate types used in the circuit. The format of each input argument will be described in detail later. The command

```
(user)% v2tau.pl <netlist>.v <netlist>.wl <model>.pm
```

produces two files: a MATLAB m-file `<netlist>_lsgs.m`, and a file describing a symbol table `<netlist>_gates`.

In MATLAB, run the m-file `<netlist>_lsgs.m` to set up the gate sizing optimization problem data for LSGS. Then run the function `lsgs` (in the LSGS toolbox) to perform gate sizing. For example:

```
>> <netlist>_lsgs.m; %define a, g, F, dmin, x0
>> x = lsgs(a, g, F, dmin, [], x0); % find optimal gate sizes
```

In this example `lsgs` produces the gate size `x` that minimizes the area with delay of the circuit less than or equal to the delay of the circuit with gate size `x0`. The gate size `x0` is a vector of gate sizes taken from `<netlist>.v`.

Alternatively, gate sizing can be performed for a timing specification `T` using the following command:

```
>> x = lsgs(a, g, F, dmin, T);
```

For complete functionality of `lsgs` see [JB07].

To back annotate the gate size `x` into the Verilog netlist, first, save `x` to a file in ASCII format (say `<netlist>_x`), using the MATLAB command:

```
>> save <netlist>_x -ascii x
```

Then use the Perl script `tau2v.pl` to perform back annotation:

```
(user)% tau2v.pl <netlist>.v <netlist>_x
```

This produces the file `<netlist>_sized.v`, a Verilog netlist with the gate sizes in `<netlist>_x` inserted for each gate. The annotated gate sizes may be decimal number, which is not in conformance with the Verilog standard. The command

```
(user)% tau2v.pl <netlist>.v <netlist>_x -snap <model>.pm
```

forces the annotated sizes to conform to the nearest allowable gate sizes as defined in the file `<model>.pm`.

The file `<netlist>_gates`, produced by `v2tau`, facilitates the back annotation performed by `tau2v`, but is not essential.

Format. Here we describe the format of the input arguments to `v2tau.pl`. The files used as examples below can be found in the `$BASE/examples` directory.

The first input argument `<netlist>.v` is a flattened, gate-level Verilog netlist. The circuit is assumed to be purely combinational with no feedback loops. We will use `ckt_and2.v` as an example. The file contains a Verilog nestlist for the circuit consisting of one nand gate driving an inverter. The type `<type>` and size `<size>` of a gate is specified by appending `<type>_<size>x` to create the module name. In this example, the instance `gate1` is of type `nand`, and the instance `gate2` is of type `inv`. The sizes of `gate1` and `gate2` are 1 and 4, respectively.

```
module ckt_and2(a, b, y);
input         a ;
input         b ;
output        y ;

  wire interconn;

  nand_1x gate1 (
     .in0             (a),
     .in1             (b),
     .out             (interconn));

  inv_4x  gate2 (
     .in              (interconn),
     .out             (y));

endmodule
```

The second input arguments `<netlist>.wl` contains the values of the wire load capacitances of the circuit. This includes the fixed load capacitances at the output of the circuit. The file `<netlist>.wl` consists of two columns. In each row the first entry is a wire label that has been instantiated in the Verilog file, and the second entry is the value of the fixed load capacitance on the wire. If a wire label is repeated, then the capacitance on the wire is taken to be the one specified by the last entry. As an example the file `ckt_and2.wl` is shown below.

```
interconn 1
y 16
```

The third input argument to `v2tau.pl`: `<model>.pm`, a Perl module, describes the attributes of each gate type. We will take `model_and2.pm` as an example. The Perl module sets up a hash of hashes data structure for each gate type. Each gate type has the following attributes which describe the RC model parameters for a unit-sized gate: the input capacitance of each input (*e.g.*, `in0` and `in1` for `nand`), the internal parasitic capacitance `internal`, and the area `area`. The `sizes` entry is the discrete array of allowed gate sizes, where the minimum allowed size must be 1.

```
our %gates =
  (
    inv => {
      inputs => {
        in => 1.0
        },
      internal => 1.0,
      area => 3.0,
      sizes => [1, 2, 4, 8]
    },

    nand => {
      inputs => {
        in0 => 1.33,
        in1 => 1.33
        },
      internal => 2.0,
      area => 6.0,
      sizes => [1, 2, 4, 8]
    },
  );
```

For each gate instantiated in the Verilog file, the input ports should correspond to the keys specified in the hash `inputs`. For example, the `nand` gate has attributes `in0` and `in1` in the hash `inputs`, and `gate1` of type `nand` has the corresponding input ports specified by `.in0` and `.in1`. Also note the script `v2tau.pl` assumes any pins not listed here are outputs.

More examples describing larger circuits and more gate types can be found in the directory `$BASE/examples`.

Notes. V2Tau was tested with Perl v5.8.8, installed on a GNU/Linux platform.

Copyright

References

[JB06] S. Joshi and S. Boyd. An efficient method for large-scale gate sizing. *Submitted to IEEE Transactions on Circuits and Systems–I: Regular Papers*, December 2006. Available at http://www.stanford.edu/~boyd/gatesizing.html.

[JB07] S. Joshi and S. Boyd. LSGS: Large-scale gate sizing MATLAB toolbox, version 0.25, March 2007. Available at http://www.stanford.edu/~boyd/lsgs.