TAU 2020 Contest
Current Source Model Delay Calculation

API Guide

v.0.1 – October 24, 2019

https://sites.google.com/view/taucontest2020

Table of Contents

Scope

This document describes technical information needed for the contestants to develop the delay calculation routines for the TAU 2020 contest.

Overview

Contestants need to develop and implement a current source model algorithm to calculate cell delays. Algorithm will be evaluated on a series of testbench circuits consisting of a driver, an interconnect, and a load, as shown in Figure 1.

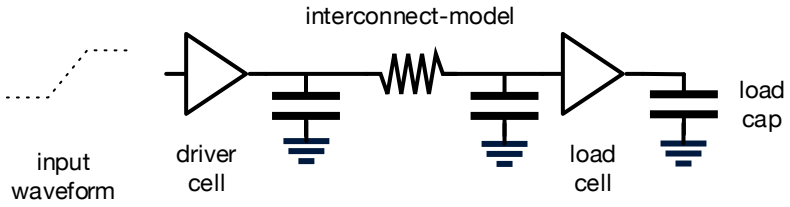

Figure 1: Test circuit for delay calculation

The infrastructure for the contest of two main programs, gen_random_nets and delay_calc_tool, interacting as shown in Figure 2 below.

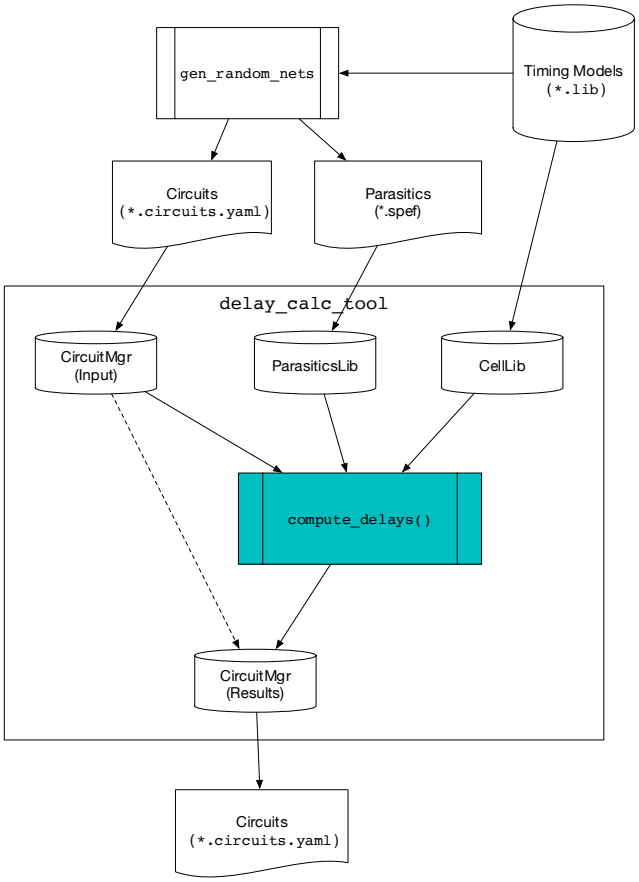

Figure 2: Infrastructure for TAU2020 Contest

gen_random_nets

`gen_random_nets` generates a set of circuits that will be run through the delay calculation program. It will read in a set of timing models, randomly select a driver cell and a load cell, and generate a random interconnect in SPEF format. The circuit is described in a YAML format. Here's an example:

```
Circuits:
  - name: net_0
    driver: I1/A/Z
    driver_celltype: CLKBUF_X3
    input_waveform: ramp 50
    driver_interconnect: 1.1 100 1.1
    load: I2/SN/Q
    load_celltype: DFFS_X1
    load_interconnect: 100 0 0
    spice_delay: 0
    spice_slew: 0
    ccs_delay: 0
    ccs_slew: 0
  - name: net_1
    driver: I5/B/ZN
    driver_celltype: OAI211_X2
    input_waveform: ramp 30
    driver_interconnect: 1.1 100 1.1
    load: I6/D/Q
    load_celltype: DFF_X1
    load_interconnect: 100 0 0
    spice_delay: 0
    spice_slew: 0
    ccs_delay: 0
    ccs_slew: 0
```

The YAML file describes a sequence of circuits defined by the following fields:

- **name**: name of the net in the SPEF
- **driver**: the instance, input pin and output pin of the driver, each separated by '/'
- **driver_celltype**: celltype of the driver
- **input_waveform**: the waveform of the driver. Currently only "ramp" is expected. The numerical value following "ramp" is the time of the ramp to transition from lower threshold to upper threshold, as defined in the Liberty model. The time unit is also as that defined in the Liberty model.
- **driver_interconnect**: a reduced, PI-model version of the interconnect, with the numbers representing cnear, resistance, and cfar, in units of ff, ohms, and ff, respectively. The full interconnect model is available in the corresponding SPEF.
- **load**: the instance, input pin and output pin of the load, each separated by '/'
- **load_celltype**: celltype of the load
- **load_interconnect**: a reduced, PI-model version of the interconnect, with the numbers representing cnear, resistance, and cfar, in units of ff, ohms, and ff, respectively. In general, we will be assuming a lumped-cap load, so resistance and cfar will both be 0.
- **spice_delay**: delay from SPICE analysis. Initialized with value 0.
- **spice_slew**: slew from SPICE analysis. Slew is defined as the time to transit from lower threshold to upper threshold, as defined in the Liberty model. Initialized with value 0.
- **ccs_delay**: delay from CCS delay calculator. Initialized with value 0.
- **ccs_slew**: slew from CCS delay calculator. Slew is defined as the time to transit from lower threshold to upper threshold, as defined in the Liberty model. Initialized with value 0.

This circuit YAML file will be accompanied by a corresponding SPEF which has the detailed parasitics. Note that the nets in the SPEF may have multiple fanout, but the load of interest is that defined in the Circuit YAML file.

delay_calc_tool

Contestants need to develop the delay calculation routine to be called by the program `delay_calc_tool`. This program will handle the following functions:

- Importing the timing models (Liberty format)
- Reading the test circuits (as shown in Figure 1)
- Reading the parasitics of the interconnect model (SPEF format)
- Calling the delay calculation (to be supplied by the contestants)
- Accumulating the delay calculation results and generating the report

Note that only the delay calculation engine is left for the contestants to write. (A default engine returning constant values will be put in place, to demonstrate the expected return values.). All other portions of the program will be developmed and distributed by the contest organizers.

`delay_calc_tool` will also be able to generate SPICE netlists and call a SPICE simulator, to generate golden results and to assist with debugging of the delay calculation routine.

Source Code

The source code is available at https://github.com/geochrist/dctk. Please read through the installation instructions, as additional packages and libraries may be needed for compilation.

The source code is still undergoing development. Please watch the repository to be notified of new updates.

Cell Libraries

The contest will be using the Nangate FreePDK45 Library. While this cell library is available without charge from the Si2 Project page at http://projects.si2.org/openeda.si2.org/projects/nangatelib, we have included a copy of the library in the dctk download for your reference. It has already been postprocessed so that it is readable by the parser we are using.

In addition, the contest will be using the ASU 7nm predictive libraries. Instructions for download are TBD.

Contest Deliverables

Contestants should submit (i) source code or linkable library that is compatible with delay_calc_tool source code running on an evaluation machine (see Section 5) and (ii) a detail document on explanation of algorithm and its implementation. Contestants are welcome to use any or none of the provided code bases.

If source code is submitted, it should be stored in src-dut/. If linkable library is delivered, please also deliver it in src-dut. It will be moved into the build/ directory (by the contest organizers) during testing and benchmarking.