Redhawk IR Drop Tool QA

+ DB Fetch and Graphing Examples

By: University of Kentucky, Introduction to VLSI

Jake Carrico, Sam McCauley, Jarren Tay, David Tran

Script Overview

irdrop.py is a QA script that analyzes a set of pgarc, cdev, spiprof files to find possible inconsistencies. In addition, it creates a database from the information in the files to allow for easy access to for querying and graphing purposes.

fetchdb.py is an interactive script to allow users to view data from the database using SQL statements.

graph.py is a script demonstrating two examples of how to use python to plot data from the database generated from irdrop.py.

How to run irdrop

We anticipate two groups of users who may wish to use our scripts. The first group may wish to run the scripts, create new QA tests, or even modify the procedurally generated database structure. This group would need to run the scripts through Python. The second group may wish to only run the scripts, without having Python and subsequent libraries installed.

Creating the file list

Irdrop.py requires a file that contains the absolute path to all of the pgarc, cdev, spiprof, and liberty files used in the QA. Each file should be separated by a new line. Files can appear in any order. An example of the filelist:

```
/users/example/redhawkOut/scf45rt.pgarc
/users/example/redhawkOut/scf45rt_PVT1.cdev
/users/example/redhawkOut/scf45rt_PVT1.spiprof
/users/example/redhawkOut/scf45rt_PVT2.spiprof
/users/example/redhawkOut/scf45rt_ssplv_typpc_fsme_tplml_PVT3.lib
```

Running through Python

Our scripts require Python 3.X, as Python 2.X has been deprecated. With a valid install of python, ensure that matplotlib and numpy are installed.

```
python3 -m pip install matplotlib --user
python3 -m pip install numpy --user
```

Afterwards, unzip the compressed folder and locate irdrop.py in the base directory. The syntax for calling the script is as follows:

```
python3 ./path/to/irdrop.py ./path/to/filelist.txt
```

Running with the distributable

The upside of the distributable is that Python and the subsequent libraries do not need to be installed. The downside is that the source code is not accessible.

The syntax for calling the script is as follows:

```
./path/to/irdropDist/irdrop/irdrop ./path/to/filelist.txt
```

Database creation

Our database schema was designed to be as similar to the files provided as possible. As such, our database has four tables, “pgarc”, “cdev”, “spiprof”, and “lib”. The data type of the columns is in parentheses.

pgarc columns:

- cell (TEXT)
- pin (TEXT)

cdev schema:

- cell (TEXT)
- temperature (REAL)
- state (TEXT)
- vector (TEXT)
- active_input (TEXT)
- active_output (TEXT)
- vpwr (REAL)
- pin (TEXT)
- esc (REAL)
- esr (REAL)
- leak (REAL)
- filename (REAL)

spiprof schema:

- cell (TEXT)
- vpwr (REAL)
- c1 (REAL)
- r (REAL)
- c2 (REAL)
- slew1 (REAL)
- slew2 (REAL)
- state (TEXT)
- vector (TEXT)
- active_input (TEXT)

- active_output (TEXT)
- pin (TEXT)
- peak (REAL)
- area (REAL)
- width (REAL)
- filename (TEXT)

lib schema:

- cell (TEXT)
- area (REAL)
- filename (TEXT)

These tables strongly resemble the structure of their representative files, so users of our QA scripts would have an easier time understanding the data structure and creating queries. A known downside of this structure is the extra capacity required to store duplicate information.

QA checks run by irdrop

- Verifying consistency between public cell name and count between cdev, spiprof, and pgarc.
- Verifying consistency of power/ground pin names
- Verifying VDD consistency between spiprof and cdev
- Verifying that in spiprof, combinational cells have 2 states and sequential cells have 4 states for each VDD variation.
- Verifying the units of parameters

Error output

The errors found are automatically outputted to “error.log” in the directory that the user called the script from. We use a set to represent the errors found. Duplicates of errors are not shown.

How to run fetchdb

We anticipate two groups of users who may wish to use our scripts. The first group may wish to run the scripts, create new QA tests, or even create new graphs. This group would need to run the scripts through Python. The second group may wish to only run the scripts, without having Python installed.

fetchdb requires irdrop to have created the redhawk.db file. If you cannot locate the redhawk.db file, please refer to the section "How to run irdrop"

Running through Python

Our scripts require Python 3.X, as Python 2.X has been deprecated. Please verify that you have python3 installed.

Afterwards, unzip the compressed folder and locate fetchdb.py in the base directory. The syntax for calling the script is as follows:

```
python3 ./path/to/fetchdb.py -d ./path/to/redhawk.db
```

The -d argument specifies the location of the database file. If not present, it searches the current directory.

Running with the distributable

The upside of the distributable is that Python and the subsequent libraries do not need to be installed. The downside is that the source code is not accessible.

The syntax for calling the script is as follows:

```
./path/to/fetchdbDist/fetchdb/fetchdb -d ./path/to/redhawk.db
```

The -d argument specifies the location of the database file. If not present, it searches the current directory.

Getting data

Our database schema was designed to be as similar to the files provided as possible. Please refer to the "Database creation" section of **How to run irdrop** to read more about the database structure and schema. Our database can be queried using SQL statements.

When prompted, enter your SQL SELECT statement, and the result will print to the terminal.

How to run graph.py

We anticipate that users who wish to view and create their own graphs will need Python to edit this script. As such, a distributable for this script has not been provided.

Running through Python

Our scripts require Python 3.X, as Python 2.X has been deprecated. With a valid install of python, ensure that matplotlib and TODO: OTHER LIBRARIES are installed.

```
python3 -m pip install matplotlib –user
python3 -m pip install numpy --user
```

Afterwards, unzip the compressed folder and locate graph.py in the base directory. The syntax for calling the script is as follows:

```
python3 ./path/to/graph.py -d ./path/to/redhawk.db
```

The -d argument specifies the location of the database file. If not present, it searches the current directory.

Graphing data examples

First, we locate the database created from irdrop.py. After this database has been located and loaded, an SQL query is performed to gather the data that we wish to plot. In this case, we are plotting VPWR vs Area, for the cell dffnrq_1x. We have multiple lines as well, one for each set of parameters tested.

To do this, we first connect to the database. Next we create and execute the SQL query:

```
SELECT DISTINCT state, vpwr, peak
FROM spiprof
WHERE cell = 'dffnrq_1x' AND pin = 'VPWR' AND c1 = 0 AND r = 0
AND c2 = 1.0e-15AND slew1 = 1.25e-11AND slew2 = 7.5e-12
AND filename LIKE '%PVT1%'
ORDER BY vpwr
```

To explain the query, we are SELECTing distinct combinations of the data values from the categories “state”, “vpwr”, and “peak”, FROM the spiprof tables, WHERE the cell name is “dffnrq_1x” and pin is “VPWR”, … and ORDER the data in ascending order by the value of VPWR.

We insert that data into an np array, which is a numpy array. These work faster than standard python lists, and it is necessary to use to graph the data with.

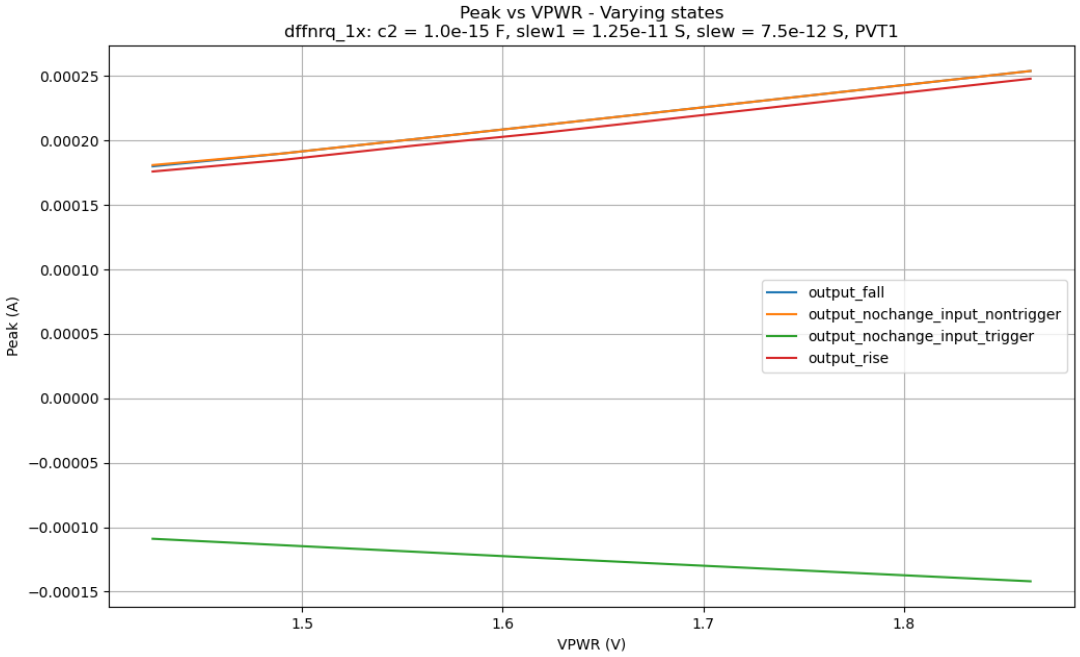
Next, we want to plot a different line for every state. To differentiate between every state we use the query:

```
SELECT DISTINCT state
FROM spiprof
ORDER BY state
```

In this query, we are SELECTing all unique values for the “state” parameter FROM the spiprof table, and we’re ordering the data in alphabetical order by the name of the state.

Finally, we create a plot for the figure. It begins as an empty plot. Then, for each state, we create a series by joining the data together from the two queries we performed. Each loop creates a series for one of the states.

Finally, we add labels to our plot and save to file.



Below is another example plot, where we plot VPWR against the area under the waveform. With this plot we are able to see how the area changes with voltage, but also how certain parameters of the simulation would also impact the area under the waveform.