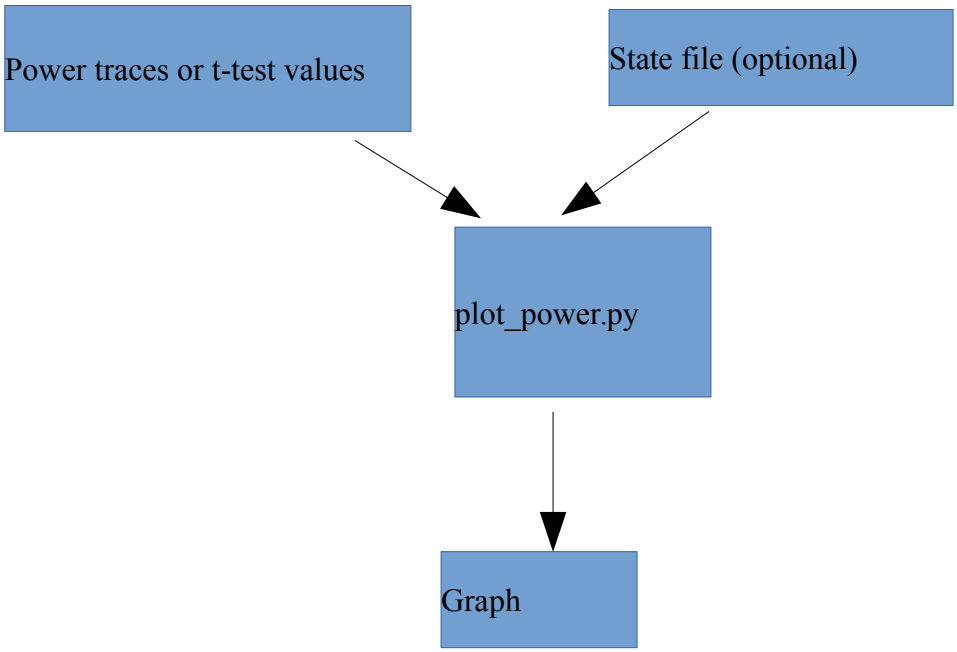
Using FOBOS Profiler

FOBOS profiler is a set of scripts that map specific time domain event (clock cycle) to the power samples in the trace collected from the oscilloscope.

Users can generate a file that maps the internal state of the DUT to clock cycles. This "state file" can be generated by modifying the test bench to write the a value corresponding to each state to a file.
The profiler script can then take this state file and the power traces and display clock transitions and the state at any clock cycle.
Other useful statistics like mean power for each clock cycle may be calculated.



Here is the usage instructions for the proiler

$ python profiler.py --help
usage: profiler.py [-h] t_values plot_file

positional arguments:
  t_values   A .npy file that store traces as Nx1 Nupmy array that contains
              t-values.
  plot_file   File name where the plot is saved

optional arguments:
  -h, --help show this help message and exit

The script expects to the state file name to be named "state_file.txt" and be at the same directory as profiler.py.

Generating the state file

Can be used to plot t-test reslut.
It can optionally:
- Plot clock signal on top of the t-plot_t_values.
- Print state numbers given state_list file

Command line arguments:

```
$ python plot_t_values.py -h
usage: plot_t_values.py [-h] t_values plot_file

positional arguments:
  t_values    A .npy file that store traces as Nx1 Nupmy array that contains
              t-values.
  plot_file   File name where the plot is saved

optional arguments:
  -h, --help  show this help message and exit
```

Also there are settings on the .py file that need to be set:

```
##########GENERAL SETTINGS
start_ylim  = -40  #Plot ymlim range
end_ylim    = 40
##########END GENERAL SETTINGS
###########CLK GRAPH SETTINGS
display_clk = 'YES' #Enable/Disable clock plotting
num_of_clks = 18   #num of clocks to in trace. Known from behavioral simulation.
clk_high    = 10   #high clock amplitude for plotting
clk_low     = -10  #low clock amplitude for plotting
state_file  = 'state_map.txt' #text file that includes states. One state in each line.
###END CLK GRAPH SETTING
```

Generating t-values : use the usual t-test2.py and add the following line to the the end of it

```
###
numpy.save("t-values.npy", numpy.array(t_array).transpose()) ##need to get one row array
####
```

Generating a state list:

For now, the period that we need to study in trace is just after di_ready becomes zero until do_valid goes to one.

in fobos_dut_tb.vhd, we add this process:

```
--Process to write states
writeState: PROCESS(clk)
  VARIABLE VectorLine: LINE;
BEGIN
IF (rising_edge(CLK)) THEN
       IF (di_ready = '0' and do_valid = '0') THEN
         hwrite(VectorLine, state_debug);
                  writeline(stateFile, VectorLine);
                    END IF;
END IF;
ASSERT False
Report "Writing States"
SEVERITY NOTE;
END Process;
--------
```

We also add these lines in architecure declarations section:

```
---
FILE stateFile: TEXT OPEN WRITE_MODE is "state_file.txt";
signal state_debug : std_logic_vector(3 downto 0);

---
```

State is assumed to be 4bit std_logic_vector reported by the victim controller (i.e FOBOS_DUT have an iterface called state_debug)
The file generated is a list of numbers each representing the state of the controller at the clock cycle identified by the line number.