CAD Programming Assignment 3

Power Analyzer

Due: 12/3 (Sun) 23:59

1. Problem Description

Given a gate-level netlist and a cell library, calculate the total power consumption by given input patterns.

There are 4 steps to follow:

Step 1 -> Build the graph according to the given netlist.
Step 2 -> Calculate the logic value, delay, and transition time.
Step 3 -> Calculate the power consumption of each gate.
Step 4 -> Calculate total power for each pattern, and toggle coverage

2. Input Format

(a) Flattened gate-level Verilog netlist (.v)

This file follows the standard Verilog gate-level syntax.
To simplify the problem, the given netlist is flattened, i.e., only one module exists in the file. Also, only three kinds of cells (NAND2, NOR2, INV) will appear in this file. Sequential circuits are not necessary to be considered.

```
module prob2(n11, n12, n13, n1, n2, n3);
       output n11, n12, n13;
       input n1, n2, n3;
       wire n4, n5, n6, n7, n8, n9, n10;

       INVX1  g1(.ZN(n4), .I(n1));
       INVX1  g2(.ZN(n5), .I(n2));
       NANDX1 g3(.ZN(n6), .A1(n4), .A2(n5));
       INVX1  g4(.ZN(n7), .I(n5));
       NOR2X1 g5(.ZN(n8), .A1(n4), .A2(n3));
       INVX1  g6(.ZN(n9), .I(n6));
       NOR2X1 g7(.ZN(n10), .A1(n6), .A2(n7));
       NANDX1 g8(.ZN(n11), .A1(n7), .A2(n8));
       NOR2X1 g9(.ZN(n12), .A1(n9), .A2(n10));
       NOR2X1 g10(.ZN(n13), .A1(n10), .A2(n8));
endmodule
```

Figure 1: An example of the gate-level netlist

(b) Input patterns (.pat)

This file gives the values of each input. The first line gives the name and order of each input. Input patterns are given in the following lines. Each pattern is given at separate line.
Note that the index of input node is not necessarily consecutive. Also, they are not necessary in ascending order. Below is a simple example for the netlist in Figure 2.
In the below example, the first pattern is n1 = 0, n2 = 1, n3 = 0.

```
input n1, n2, n3
0 1 0
1 1 0
1 1 1
.end
```

Figure 2: An example of the input pattern

(c) **Simplified liberty file (test_lib.lib)**

This file is simplified from the standard library format, which collects the timing and power information for each cell. The default units are also given in the file, which are ns (timing), pF (capacitance), V (voltage), and mA (current).

To simplify the problem, we assume that the timing and power information of the same output is the same for all input path, i.e., A1->ZN and A2->ZN have the same delay and power. Therefore, only one table is recorded for each output.

The table index is defined in lu_table_template, which include total output loading and input transition time. Please note that the total output loading and input transition time should be calculated based on the real circuit connection. The output rising/falling time of a cell will be the input transition time of the succeeding cells. The output loading is the total input capacitance for all fanout gates.

Please note that the values in the liberty file may not be the same in the given .lib file. But TA will always use the same format of .lib file given to you to verify your program.

In Figure 3, the input of A1 capacitance for NOR gate is 0.0105008 pF, and the input of A2 capacitance for NOR gate is 0.0108106pF.

```
cell (NOR2X1) {
  pin(A1) {
    direction : input;
    capacitance : 0.0105008;
  }
  pin(A2) {
    direction : input;
    capacitance : 0.0108106;
  }
…
```

Figure 3: A simplified example of liberty file

In Figure 4, when the output logic value is 1 for NOR2, we should refer to this rise power table. The table index is defined in Figure 5, the horizontal axis is total output loading, and the vertical axis is input transition time.

```
Cell (NOR2X1)
…
rise_power(table10){
        values("0.010451,0.012200,0.010200,0.011217,0.012527,0.015876,0.022985",\
               "0.012401,0.012343,0.010570,0.010892,0.012249,0.016267,0.022506",\
               "0.007618,0.009438,0.008822,0.010576,0.011355,0.017262,0.021875",\
               "0.011155,0.011529,0.007943,0.009965,0.011008,0.013510,0.021533",\
               "0.010841,0.009993,0.009287,0.009466,0.008619,0.012171,0.019587",\
               "0.010153,0.009880,0.009047,0.007202,0.005313,0.007439,0.011703",\
               "0.010490,0.010080,0.009190,0.007640,0.004860,0.000720,0.003490");
      }
…
```

Figure 4: Rise power table in liberty file

```
lu_table_template(table10){
  variable_1 : total_output_net_capacitance;
  variable_2 : input_transition_time;
  index_1 ("0.001400,0.003000,0.006200,0.012500,0.025100,0.050400,0.101000");
  index_2 ("0.0208,0.0336,0.06,0.1112,0.2136,0.4192,0.8304");
}…
```

Figure 5: Table index in liberty file

3. Output Format

After executing your program, four output files should be generated for each step. You should follow the naming rules as below. For example, if the file name of the Verilog netlist file is "*c17.v*", the <case_name> would be "c17".

Step 1: *<student_id>_<case_name>_load.txt*
Step 2: *<student_id>_<case_name>_gate_info.txt*
Step 3: *<student_id>_<case_name>_gate_power.txt*
Step 4: *<student_id>_<case_name>_coverage.txt*

(a) Step 1: Calculate output loading of each gate

First, construct the circuit graph according to the given netlist. In this step, we proceed to verify the correctness of the graph construction by checking the output loading of each cell. The output loading of each cell would be the summation of the input capacitance of all the fanout cells. For the primary output in the given netlist, please set the output loading as 0.03pF.

Store yore results in "***<student_id>_<case_name>_load.txt***" and sort your results according to instance number in ascending order.

Besides, you must print each value to the 6th decimal places.

```
g1 0.030000
g2 0.018106
g3 0.010500
……
g8 0.030000
g9 0.007984
g10 0.018896
```

Figure 6: Example of load.txt

(b) Step 2: Calculate logic output, cell delay, transition time of each gate

1. Logic output

Calculate the output logic value of each gate according to the input pattern.

2. Cell delay, transition time

After you get the logic output of each gate, look up the table according to input transition time and output capacitance. For input gate, the input transition time is 0ns.

For the input transition time, you must choose which input to use as the input transition time, which is related to the controlling value and the total delay. For sensitization rule, please refer to note (g) for more details.

Sort your results according to instance number in ascending order and set the file name as "***<student_id>_<case_name>_gate_info.txt***".

```
g1 1 0.052093 0.079336
// output=1, output rise delay = 0.052093, output rise transition time = 0.079336
g2 1 0.070989 0.118823
g3 1 0.066306 0.108485
g4 1 0.043873 0.063469
g5 0 0.068176 0.111416
// output=0, output fall delay = 0.068176, output fall transition time = 0.111416
……
```

Figure 7: Example of gate_info.txt

If there are more than one input patterns, please report the delay information of each input pattern separated by a blank line.

```
g1 (logic value) (cell delay) (transition time)
// the beginning of the first pattern
…
g10 (logic value) (cell delay) (transition time)
// blank line here !!
g1 (logic value) (cell delay) (transition time)
// the beginning of the second pattern
…
g10 (logic value) (cell delay) (transition time)
```

Figure 8: Format of gate_info.txt

(c) **Step 3: Calculate internal power and switching power of each gate.**

1. **Internal power**

With the input transition time and output capacitance, we can look up the power table from liberty file to get the internal power of each gate. For each gate, it must consume internal power for each pattern.

2. **Switching power**

Switching power is related to the toggle rate of each gate. We define that when data toggles from 1 to 0 or from 0 to 1, it always consumes $\frac{1}{2}CV^2$ power. (C is the output capacitance of each gate, and we define the **V** as 0.9V in this problem. If there is no data transition for the gate, then the switching power will be 0, but you should assume each gate will toggle in this step.

Sort your results according to instance number in ascending order and set the file name as "***<student_id>_<case_name>_gate_power.txt***".

```
g1 0.007962 0.040832
// output rise power = 0.007962, switching power = 0.040832
g2 0.011845 0.015369
g3 0.008915 0.007738
g4 0.000541 0.027317
g5 0.011627 0.009812
// output=0, output fall power = 0.011627, switching power = 0.009812
……
```

Figure 9: Example of gate_power.txt

If there are more than one input patterns, please report the delay information of each input pattern separated by a blank line.

```
g1 (Internal power) (Switching power)
// the beginning of the first pattern
…
g10 (Internal power) (Switching power)
// blank line here !!
g1 (Internal power) (Switching power)
// the beginning of the second pattern
…
g10 (Internal power) (Switching power)
```

Figure 10: Format of gate_power.txt

(d) **Step 4: Calculate total power for each pattern, and toggle coverage.**

1. Total power

In this step, you need to add up the internal power of all the gates and calculate the switching power according to the data transition. If the gate has no data transition, then the switching power consumption is 0.

2. Toggle coverage

In this part, we only check the toggle coverage, which means we test whether all the gates have toggled from 0 to 1 and from 1 to 0 for 20 times each. Then calculate the final coverage percent.

If there are only two gates in the whole circuit, and if after all the patterns, the toggle times of g5 and g6 is like below table, then the coverage will be 53/80 = 66.25 %

Table 1: Example of toggle coverage calculation

	Toggle from 0 to 1	Toggle from 1 to 0
Gate g1	35	5
Gate g2	8	20
Toggle coverage	66.25%	

The file name is "<***student_id***>_<***case_name***>_***coverage.txt***"

You must print the index of input pattern (start from 1)**,** and total power consumption to the 6th decimal places and the toggle coverage to the 2nd decimal places.

Note that the toggle coverage is a cumulative value, but the total power consumption only depends on the current pattern.

```
1 0.556921 2.78%
// blank line here !!
2 0.437895 5.00%
```

Figure 11: Example of total_power.txt

```
(# of Patterns) (Total power consumption) (percentage of output coverage)
// blank line here !!
(# of Patterns) (Total power consumption) (percentage of output coverage)
```

Figure 12: Format of total_power.txt

4. Notice

(a) The delay time of each instance can be determined by the input transition time (the output transition time of its driving cell) and the output loading (the total input capacitance for all fanout gates) according to the given delay table.

Boundary Conditions:

- Set input transition time of each primary input as **0ns.**
- Set output loading of each primary output as **0.03pF.**
- **Initial logic value of each gate is 0.**

(b) To simplify the problem, there are several assumptions below.

- While calculating the logic output of each instance, assume it is in **zero-delay mode**. In other words, please ignore the glitch during logic transition and use the final value directly. You can get the input values from the outputs of your driving gates.
- If there are multiple paths from cell inputs to cell output, please use the **sensitization rules** to find the **sensitizable path** and use it to determine the delay value.

- For the delay time, there are two tables (cell_rise, cell_fall).
  If the output value of this cell is 1, use the table cell_rise.
  If the output value of this cell is 0, use the table cell_fall.
- For the output transition time, there are also two tables (rise_transition, fall_transition).
  If the output value of this cell is 1, use the table rise_transition.
  If the output value of this cell is 0, use the table fall_transition.

  The successor cells will use this value as the input transition time.

(c) Print all output value to the 6th decimal places except output logic value of the gate, the pattern index, and the toggle coverage.

(d) If the result of the extrapolation is a negative value, please set it to 0.

(e) **No wire delay** in the HW3.

(f) **There is no need to handle the comment (// 、/**/) in HW3**

(g) **Path sensitization rule**:

- Rule 1: Input transition time is determined by the driving cell which has the earliest delay of controlling value.
- Rule 2: Input transition time is determined by the driving cell which has the latest delay of non-controlling value.

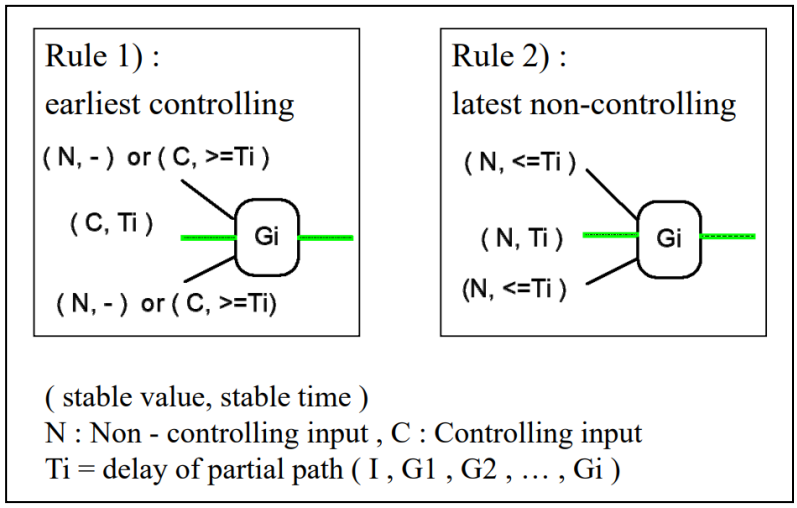

Figure 13: path sensitization rule

(h) The values in the test_lib.lib will be changed. But the format will be the same.

(i) The margin of error will be | your answer – golden answer | / (golden answer) < 0.001%

5. Compile and Execution

- You must include "*Makefile*" for TA to compile your code.
- Name of the binary after compiling must be <*student_id*>
- Your program must be able to receive commands in following format.

  $./<student_id> netlist_file input_pattern test_lib.lib

  For example:

  $./311510369 c17.v c17.pat test_lib.lib

- Be sure that it can compile and execute successfully on our workstation.
- Do not print any words on terminal when executing.

6. Grading Policy

You are encouraged to complete this assignment step by step.
For each testcase, you would get your grade for each step with corresponding percentage of the total grade as shown below. The remaining 20% not included for the correctness of each step would be rated by the ranking of runtime among all students who pass all four steps. Please note that you would not get any grade for the runtime performance unless you pass all four steps.

(a) Correctness of step 1 result (20%)
(b) Correctness of step 2 result (20%)
(c) Correctness of step 3 result (20%)
(d) Correctness of step 4 result (20%)
(e) Runtime performance (20%)

Hidden cases will be evaluated.

* Violates file naming rules: -10 points
* Doesn't follow the requirements of output format: -20 points
* Print words on terminal while executing: -20 points
* Plagiarism: -100 points

7. Submission Steps

(a) Put these files in a folder, the name of the folder is your student_id.
- Source code(.cpp, .h)
- Makefile
- A simple report that explains the implementation details, name the report as "*<student_id>_HW3_report.pdf*"

(b) Use the command below to compress the folder in linux environment, then compressed file name should also be your student_id.

```
$tar cvf <student_id>.tar <student_id>
```

(c) Submit <student_id>.tar to E3 before the deadline.

* If you have any question, please ask on HW3 discussion forum of E3.