Practice 12 V1.0 (2015/12/29)

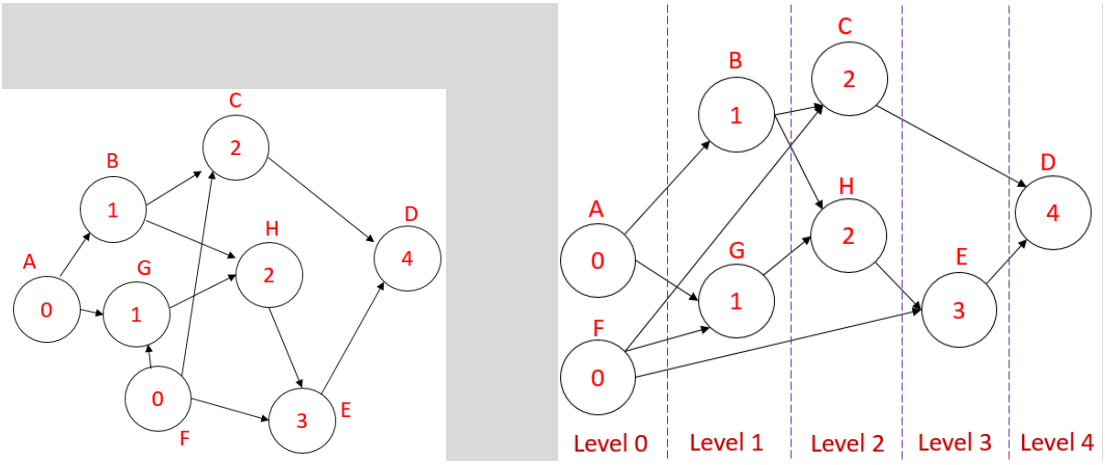
Logic simulator

1. Directed graph → Leveled directed graph by causality

A. **A graph consists of vertices and edges. A graph is a directed graph if each edge has specific direction. The edge direction implies the causality of something represented in the graph. Consider the causality of a directed graph, we can re-arrange the vertices of a directed graph in a causality order.**

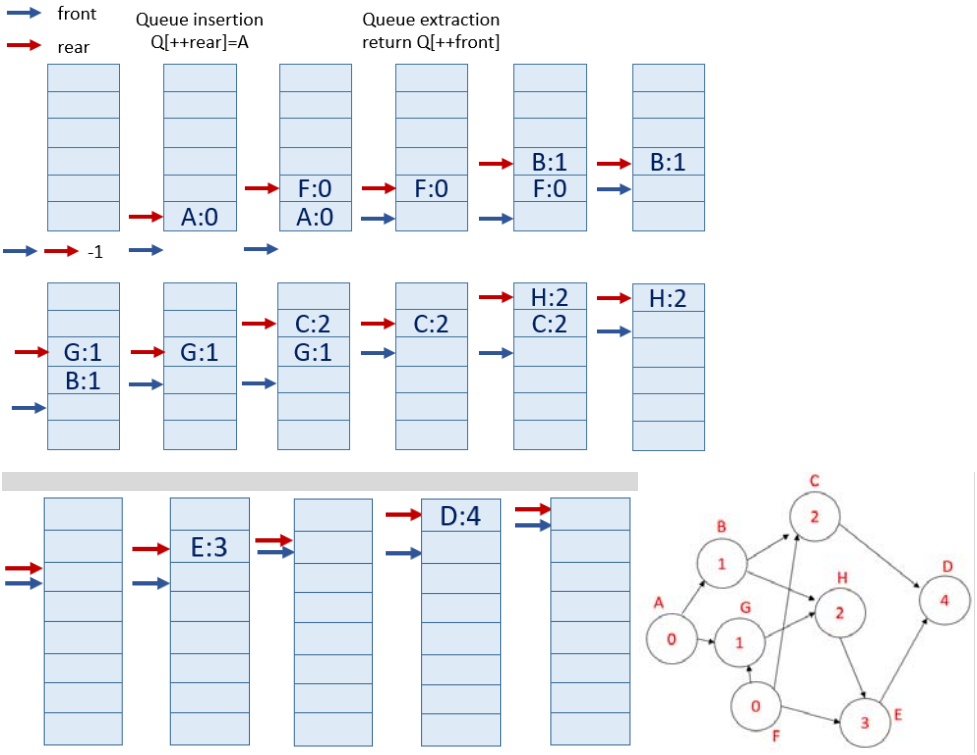
B. **A graph is acyclic if it has no cycle. A directed graph is acyclic if it has no directed cycle. If we map a logic gate of a combinational circuit as a vertex in a directed graph and map a connection between two logic gates as a directed edge connecting two associated vertices of these two logic gates, the graph is acyclic.**

C. **After leveling an acyclic directed graph, we can compute the logic simulation on the graph according its causality order.**



D. Example

```
1.  Put all vertices of 0 in-degree into a queue
2.  while (queue is not empty)          {
3.      extract one vertex v from the queue;
4.      for each vertex u pointed by v  {
5.          increase visited number of u by 1;
6.          if (visited number == fanin number)  {
7.              level number = level(v) + 1;
8.              put vertex u into queue;
9.          } // end if
10.     } // end for
11. } // end while
```



2. Problem

Logic simulator is a very important simulation technique to verify the correctness of a circuit design. Designers often apply logic simulator to his designs to obtain their output response based on a given set of input patterns. After that the output responses are compared with a golden set of outputs to check if the designs are correct. The problem is pretty simple but huge problem instance size complicates the implementation of methods to realize logic simulation. Since to apply all combinations of input patterns to the circuit under test is impossible, we are usually given a set of input patterns and then apply all input patterns to logic simulator. The way to put a input pattern into an integer is simple for implementation but inefficient for the performance of logic simulator. Thus generally several input patterns are packed into an integer, such as 32 patterns for a machine of 32-bit word length. With this scheme, we can perform logic simulation for 32 input patterns at a time.

Input format:

Circuit input file:

Signal_no Signal_name gate_type fanout_no fanin_no other

If fanout_no is larger than 1, the following lines describe fanout

Circuit input example:

```
3       3gat inpt       2       0 >sa0 >sa1
8       8fan from       3gat            >sa1
9       9fan from       3gat            >sa1
```

```
 10      10gat nand      1     2         >sa1
   1       8

 11      11gat nand      2     2 >sa0 >sa1
  9       6
 14      14fan from      11gat         >sa1
 15      15fan from      11gat         >sa1
 22      22gat nand      0     2 >sa0 >sa1
    10      20
```

```
*c17 iscas example (to test conversion program only)
*-----------------------------------------------
*
*
*  total number of lines in the netlist ..............    17
*  simplistically reduced equivalent fault set size =     22
*       lines from primary input  gates .......      5
*       lines from primary output gates .......      2
*       lines from interior gate outputs ......      4
*       lines from **     3 ** fanout stems ...      6
*
*       avg_fanin  =  2.00,     max_fanin  =  2
*       avg_fanout =  2.00,     max_fanout =  2
*
*
    1     1gat inpt    1   0       >sa1
    2     2gat inpt    1   0       >sa1
    3     3gat inpt    2   0 >sa0 >sa1
    8     8fan from     3gat       >sa1
    9     9fan from     3gat       >sa1
    6     6gat inpt    1   0       >sa1
    7     7gat inpt    1   0       >sa1
   10    10gat nand    1   2       >sa1
     1     8
   11    11gat nand    2   2 >sa0 >sa1
     9     6
```

```
   14    14fan from    11gat       >sa1
   15    15fan from    11gat       >sa1
   16    16gat nand    2   2 >sa0 >sa1
     2    14
   20    20fan from    16gat       >sa1
   21    21fan from    16gat       >sa1
   19    19gat nand    1   2       >sa1
    15     7
   22    22gat nand    0   2 >sa0 >sa1
```

Input pattern file: For the circuit whose input number is below 20, the input pattern file will not be given. You have to apply every pattern to logic simulation and print the output responses in increasing order of pattern number. For the circuit whose input number is above 20, a input pattern file with 1 million input patterns will be offered

Pattern 1

Pattern 2

…

Each pattern is an unsigned long integer and is printed at a line. For the circuit

Input pattern example:

8474831

299393213

…

Output format:

In_no1 In_no2 … In_noN || Out_no1 Out_no2 … Out_noK

The input and output signals are printed in decreasing order of their signal number. Two adjacent signals are separated by one space character.

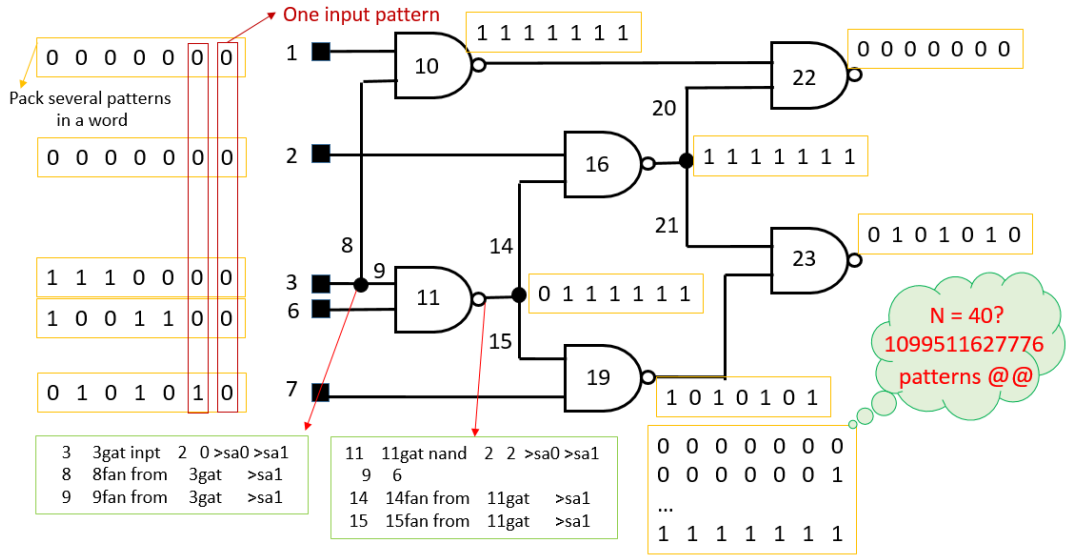
Output example:

7 6 3 2 1 || 23 22

0 0 0 0 0 || 0 0

0 0 0 0 1 || 0 0

0 0 0 1 0 || 1 1



Main features:

(a) Given a circuit file and an input pattern file (option). Construct leveled directed graph and the gates of the same level are stored in a level list. Print the level list (due tonight).

(b) Pack every 32 input patterns in a word and perform 32 input-pattern logic simulation on the graph at a time. Write input pattern and output responses to output file cxxx.out. For instance, you have to generate a output file names as c17.out for the circuit c17.isc (due in 1/14).