Instructions

In EE658 you will be required to design/implement some logic and circuit simulators and test generators. A small C program is provided to form the basis for your programs.

See the following two files included in "**program.tar.gz**":

1) **readckt.c**: reads circuit netlists in "self" format (will be described later) into the data structrures (will be described later)
2) **Makefile**: contains the information for compiling multiple source files together to create an executable file in UNIX system.

Three example circuits (in "**circuit.tar.gz**") are provided in the "self" format:

add2.ckt, **c17.ckt**, and **x3mult.ckt**

To compile the program run:

```
make
```

To execute the program run: (you can find an example on next page)

```
readckt
```

Also, in "**self_format.pdf**", you can find the specification of the "self" format, followed by a circuit representation using this format.

In this format, we assume that each primary input is the output of an imaginary gate called **input gate**, each fan-out branch is the output of an imaginary gate called **fan-out gate**, and we call each gate (real or imaginary) together with its output line **node**.

In the example in "**self_format.pdf**", you can see that each row carries all the required information for each node (from node 1 to 14). For instance, in row 13, the "self" format representation is **0 13 3 1 2 8 11**. In specific, **0** means it's only a gate output, **13** means the outline is 13, **3** means it's an OR gate, **1** means it has 1 fanout, **2** means it has 2 fanins, **8** and **11** are those two fanins.

Note: **PI** - Primary Input; **FB** – Fanout Branch; **PO** – Primary Output; **IPT** – Input; **BRCH** - Branch

You can also find the description of the "self" format in the comments of "readckt.c".

Another PDF - "**data_structure.pdf**" shows the data structure we use here to represent circuit. You need to understand this representation so that you can write your own part of code by using this data structure.

Example for using readckt:

To read in circuit file c17.ckt run:

```
Command>read ../circuit/c17.ckt
```

The screen will show

```
==> OK
```

To print circuit information of recently-read circuit file, say c17.ckt, run:

```
Command>pc
```

The screen will show

```
Node   Type        In          Out
------ ------      -------     -------
     1  PI                      10
     2  PI                      16
     3  PI                      8 9
     8  BRANCH   3              10
     9  BRANCH   3              11
     6  PI                      11
     7  PI                      19
    10  NAND       1 8          22
    11  NAND       9 6          14 15
    14  BRANCH  11              16
    15  BRANCH  11              19
    16  NAND       2 14         20 21
    20  BRANCH  16              22
    21  BRANCH  16              23
    19  NAND       15 7         23
    22  NAND       10 20
    23  NAND       21 19
Primary inputs:  1 2 3 6 7
Primary outputs: 22 23

Number of nodes = 17
Number of primary inputs = 5
Number of primary outputs = 2
```

To print help information run:

```
Command>help
```

The screen will show

```
READ filename - read in circuit file and creat all data structures
PC - print circuit information
HELP - print this help information
QUIT - stop and exit
```

To exit the program run:

```
Command>exit
```

The program will be terminated and return to system command prompt.