CAD for VLSI Design

Project Assignment 1

Benchmark Translator

Instructor: Andy, Yu-Guang Chen Ph.D.

TA: Yi-Ting Lin

Department/Class: Electrical Engineering 4A

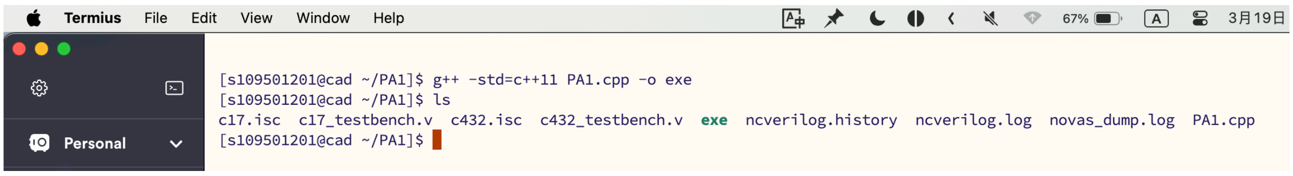
Name: **陳緯亭**

Student ID Number: **109501201**

Contents

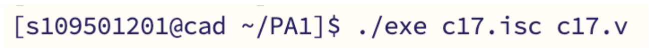
List Of Listings

1. How I compile and execute the program



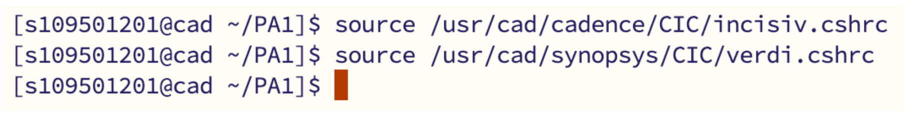
```
[s109501201@cad ~/PA1]$ g++ -std=c++11 PA1.cpp -o exe
[s109501201@cad ~/PA1]$ ls
c17.isc  c17_testbench.v  c432.isc  c432_testbench.v  exe  ncverilog.history  ncverilog.log  novas_dump.log  PA1.cpp
[s109501201@cad ~/PA1]$
```

Fig 1: Generate an executable file



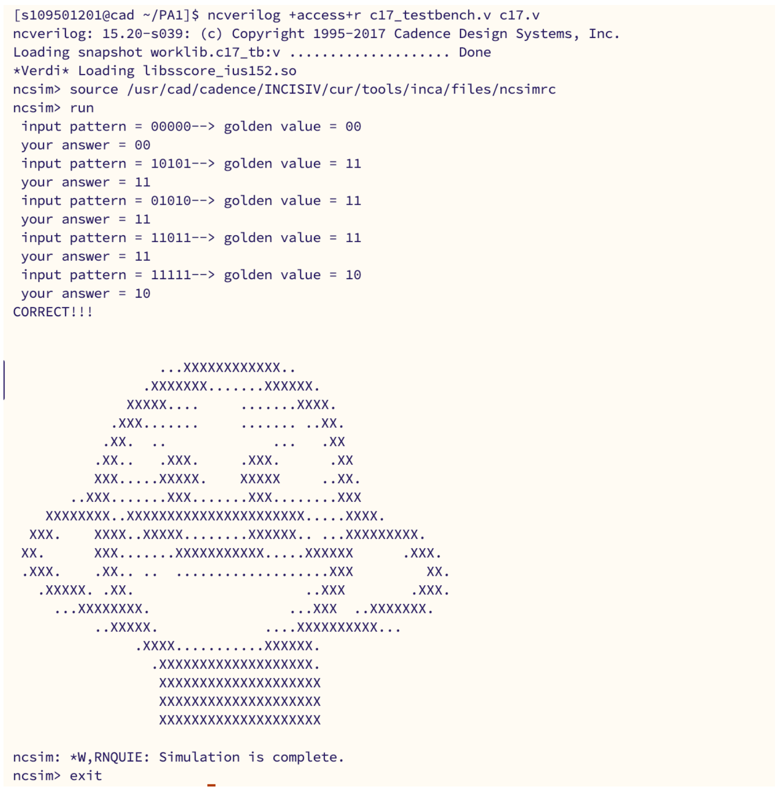
```
[s109501201@cad ~/PA1]$ ./exe c17.isc c17.v
```

Fig 2: Use the executable file to ISCAS’85 netlist into Verilog format



```
[s109501201@cad ~/PA1]$ source /usr/cad/cadence/CIC/incisiv.cshrc
[s109501201@cad ~/PA1]$ source /usr/cad/synopsys/CIC/verdi.cshrc
[s109501201@cad ~/PA1]$
```

Fig 3: Source the following commands



```
[s109501201@cad ~/PA1]$ ncverilog +access+r c17_testbench.v c17.v
ncverilog: 15.20-s039: (c) Copyright 1995-2017 Cadence Design Systems, Inc.
Loading snapshot worklib.c17_tb:v .................... Done
*Verdi* Loading libsscore_ius152.so
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
 input pattern = 00000--> golden value = 00
 your answer = 00
 input pattern = 10101--> golden value = 11
 your answer = 11
 input pattern = 01010--> golden value = 11
 your answer = 11
 input pattern = 11011--> golden value = 11
 your answer = 11
 input pattern = 11111--> golden value = 10
 your answer = 10
CORRECT!!!


                  ...XXXXXXXXXXXX..
                .XXXXXXX......XXXXXX.
              XXXXX....     ......XXXX.
            .XXX.......     ....... ..XX.
           .XX.  ..             ...   .XX
          .XX..   .XXX.     .XXX.      .XX
          XXX.....XXXXX.    XXXXX     ..XX.
       ..XXX.......XXX.......XXX........XXX
    XXXXXXXX..XXXXXXXXXXXXXXXXXXXXXX.....XXXX.
  XXX.    XXXX..XXXXX........XXXXXX.. ...XXXXXXXXX.
 XX.      XXX.......XXXXXXXXXXX.....XXXXXX      .XXX.
 .XXX.    .XX.. ..  ...................XXX         XX.
   .XXXXX. .XX.                      ..XXX       .XXX.
     ...XXXXXXXX.                 ...XXX  ..XXXXXXX.
          ..XXXXX.             ....XXXXXXXXXX...
               .XXXX...........XXXXXX.
                 .XXXXXXXXXXXXXXXXXXX.
                  XXXXXXXXXXXXXXXXXXXX
                  XXXXXXXXXXXXXXXXXXXX
                  XXXXXXXXXXXXXXXXXXXX

ncsim: *W,RNQUIE: Simulation is complete.
ncsim> exit
```

Fig 4: Check the correctness

2. Pseudo Code

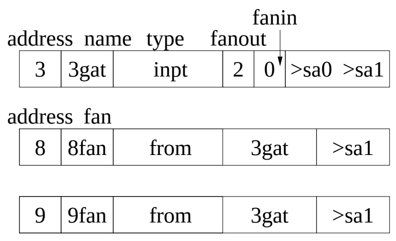


Fig 5: The Variables store the information

Algorithm 1 Output Algorithm

```
function output_file($vFile$: char*, $c$: PCKT):        ▷ O($numNode^2 * numFanin$)
    ofstream $outFile(vFile)$;
    \\ module to $outFile$
    \\ input to $outFile$
    \\ output to $outFile$
    \\ wire to $outFile$
    for $i = 0$ to $c \to numNode$ do
        if $c \to nodes[i] \to type \neq INPT$ and $c \to nodes[i] \to type \neq FROM$ then
            \\ type and instance to $outFile$
            for $j = 0$ to $c \to nodes[i] \to numFanin$ do
                $k \leftarrow 0$
                while $k < c \to numNode$ do
                    if $c \to nodes[k] \to address == c \to nodes[i] \to fanins[j]$ then
                        Write ", " + $c \to nodes[k] \to outname$ to $outFile$        ▷ Search form the beginning address.
                    end if
                    $k \leftarrow k + 1$
                end while
            end for
        end if
    end for
end function
```

Algorithm 2 Second Pass Algorithm

```
function second_pass($inFile$: ifstream, $c$: PCKT):
    string $s$
    int $address, fanout, fanin$
    char $name[256], type[256]$
    int $col \leftarrow 0$
```

```
 6:     while getline($inFile$, $s$) do
 7:         istringstream $iss(s)$
 8:         if $s[0] ==' *'$ then
 9:             continue
10:         end if
11:         $iss \rightarrow address, name, type, fanout, fanin$
12:         \\ Get the information from $inFile$ and Store in $c$
13:         if fanin $> 0$ then
14:             for $i = 0$ to $fanin$ do
15:                 $iss \rightarrow c \rightarrow$ nodes$[col] \rightarrow$ fanins$[i]$
16:             end for
17:         end if
18:         if fanin $\neq 0$ then
19:             $c \rightarrow$ nodes$[col] \rightarrow$ type $\leftarrow$ DRIVER
20:             $c \rightarrow$ nodes$[col] \rightarrow$ outname $\leftarrow$ "$gat\_out$" $+ address$
21:             if fanout $== 0$ then
22:                 $c \rightarrow$ nodes$[col] \rightarrow$ type $\leftarrow$ OUTPT
23:             end if
24:         end if
25:         $col \leftarrow col + 1$
26:         if fanout $> 1$ then
27:             for $i = 0$ to $fanout$ do
28:                 getline($inFile$, $s$)
29:                 istringstream $iss(s)$
30:                 $iss \rightarrow address, fan$
31:                 \\ Get the information from $inFile$ and Store in $c$
32:                 $col \leftarrow col + 1$
33:             end for
34:         end if
35:         $c \rightarrow$ numNode $\leftarrow col$
36:     end while
37:     return $c$
38: end function
```

3. PA

3.1 The degree of completion of the assignment: ALL

3.2 Code Explanation

Listing 1: Preprocessor

```cpp
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include <string.h>
```

According to Listing 2, there are to structure here to keep all the neccessary circuit information in advantage of facilitate information transfer.

Listing 2: Struct and Function prototype

```cpp
using namespace std;

/* Define the the function name performed by the gate driving this node.*/
enum GATENAME {
    INPT, FROM, OUTPT, DRIVER
};

/* Information about a single node in the circuit. */
struct NODE{
    int address;   // A unique number that differentiates the node.
    char* name;    // Provide meaningful information about the node usage.
    GATENAME type; // The function.
    int numFanin;  // The number of gates.
    int* fanins;   // Array for the fanin.
    char* func;    // Name of the function
    char *outname;  // Port name to ofstream into the .v file.
};

typedef struct NODE *PNODE;

struct CKT{
    PNODE *nodes;     // Array of all node
    char* name;       // Name of the CKT
    int  numNode;     // The total node to form the CKT.
    int numInput;     // Number of total input wires
    int numOutput;    // Number of total output wires
    int numDriver;    // Number of total driver wires
};
```

```
typedef struct CKT *PCKT;


PCKT first_pass(ifstream& inFile, PCKT c);
PCKT second_pass(ifstream& inFile, PCKT c);
PCKT input_file(char *iscFile);
void output_file(char *vFile, PCKT c);
```

According to Listing 3, I seperate the behavior into two branches - input_file and output_file, and then free the memory before the program finish.

Listing 3: The main function

```
int main(int argc, char *argv[]){
    PCKT c;

    c = input_file(argv[1]);

    output_file(argv[2], c);

    free(c);

    return 0;

}
```

According to Listing 4, the ISCAS'85 format file is read twice. The first time is to collect certain information from the file that will later be used to allocate memory. Subsequently, the information will be thoroughly collected during the second pass. (Note: the file pointer should be moved to the beginning) Since the file name is crucial, as it contains information that will be instantiated as a module with the same name.

Listing 4: Input file process

```
PCKT input_file(char *iscFile){

    PCKT c;
    ifstream inFile(iscFile);

    c = (PCKT) malloc (sizeof(CKT));

    if(!inFile){
        cout << "isc file can't not be open.";
    }

    c->name = iscFile;
```

```cpp

    c = first_pass(inFile, c);


    inFile.clear(); // Clear any error flags
    inFile.seekg(0, ios::beg); // Reset file pointer to beginning

    c = second_pass(inFile, c);

    cout << "NO " << c -> nodes[0] -> name << endl;

    inFile.close(); // Close the file

    return c;
}
```

According to Listing 5, the Verilog file will be generated here. The output stream should begin with the unique module name followed by its port interface list. Subsequently, the output stream should include input signal and output definitions. Next, there should be net type declarations and their instances, followed by output ports and their corresponding input port lists.

Listing 5: Output file process

```cpp
void output_file(char *vFile, PCKT c){

    ofstream outFile(vFile);

    if(!outFile){
        cout << "v can't be generate.";
    }

    outFile << "`timescale 1ns/1ps\n";

    bool notFirst = 0;

    // module
    outFile << "module " << string(c->name).substr(0, string(c->name).length() - 4) <<
         " (";
    for(int i = 0; i < c-> numNode; i++){
        if(c -> nodes[i] -> type == INPT || c -> nodes[i] -> type == OUTPT){
            if(notFirst)
                outFile << ", ";
            outFile << c -> nodes[i] -> outname;
            notFirst = 1;
        }
    }
    outFile << ");\n";

    // input
    notFirst = 0;
    outFile << "input ";
    for(int i = 0; i < c->numNode; i++){
        if(c -> nodes[i] -> type == INPT){
            if(notFirst)
                outFile << ", ";
            outFile << c -> nodes[i] -> outname;
            notFirst = 1;
        }
    }
    outFile << ";\n";

    // outputs
    notFirst = 0;
    outFile << "output ";
    for(int i = 0; i < c->numNode; i++){
        if(c -> nodes[i] -> type == OUTPT){
            if(notFirst)
                outFile << ", ";
            outFile << c -> nodes[i] -> outname;
            notFirst = 1;
        }
    }
    outFile << ";\n";

    // wire
    if(c -> numDriver > 0){
        notFirst = 0;
        outFile << "wire ";
        notFirst = 0;
        for(int i = 0; i < c -> numNode; i++){

            if(c -> nodes[i] -> type == DRIVER){
                if(notFirst)
                    outFile << ", ";
                outFile << c -> nodes[i] -> outname;
                notFirst = 1;
            }
        }
            outFile << ";\n\n";
    }

    outFile << "\n";

    for(int i = 0; i < c -> numNode; i++){
        if(c -> nodes[i] -> type != INPT
        && c -> nodes[i] -> type != FROM){
            outFile << c -> nodes[i] -> func << " ";
            outFile << c -> nodes[i] -> func << c -> nodes[i] -> address << " ( ";
            outFile << c -> nodes[i] -> outname ;
            for(int j = 0; j < c -> nodes[i] -> numFanin; j++)
            {
                int k = 0;
                while(k < c -> numNode){
                    if (c->nodes[k]->address == c->nodes[i]->fanins[j]) {
                            outFile << ", " << c->nodes[k]-> outname;
                    }
                    k++;
                }
            }
            outFile << " );\n";
        }
    }
    outFile << "\nendmodule\n";
}
```

According to Listing 6, the purpose is to gather preliminary information from the node line to facilitate memory allocation.

Listing 6: First file process

```
PCKT first_pass(ifstream& inFile, PCKT c){
    string s;
    int address, fanout, fanin;
    string name, type;
    int numOutput = 0, numInput = 0;
    int maxAddress = -1;
    int lineCount = 0;                  // column line cause by number of fanout

    while(getline(inFile, s)){
        istringstream iss(s);
        if(s[0] == '*') continue;
        iss >> address >> name >> type >> fanout >> fanin;
        lineCount += fanout;
        maxAddress = (address > maxAddress) ? address : maxAddress;
        if(fanin > 0) getline(inFile, s);
        if(fanout > 1){
            while(fanout-- > 0){
                getline(inFile, s);
            }
        }
        if(fanout == 0) numOutput++;
        if(fanin == 0)  numInput++;
    }

    c -> numOutput = numOutput;
    c -> numInput = numInput;
    c -> nodes = (PNODE *) malloc ((maxAddress + lineCount + 8) * sizeof(PNODE));
    cout << "LineCount: " << lineCount << endl;
    cout << "maxAddress: " << maxAddress << endl << endl;

    return c;
}
```

According to Listing 7, the information should be stored in the structure in this time. To notice that the space should be store the fanin addresses, so it need to be allocate memory.

Listing 7: Second file process

```
PCKT second_pass(ifstream& inFile, PCKT c){

    string s;
    int address, fanout, fanin;
    char name[256], type[256];
    int col = 0;

    while(getline(inFile, s)){
        istringstream iss(s);
        if(s[0] == '*') continue;
        iss >> address >> name >> type >> fanout >> fanin;
        cout << address << " " << name << " " << type << " " << fanout << " " << fanin
             << endl;

        stringstream add;
            add << address;

            string fullname = "gat" + add.str();

            c->nodes[col] = (NODE *)malloc(sizeof(NODE));
            c->nodes[col]->address = address;
            c->nodes[col]->name = strdup(name);
            c->nodes[col]->type = INPT;
            c->nodes[col]->numFanin = fanin;
            c->nodes[col]->fanins = (int *)malloc(fanin * sizeof(int));
            c->nodes[col]->func = strdup(type);
            if(strcmp(type, "buff") == 0)
                c->nodes[col]->func = strdup("buf");
            c->nodes[col]-> outname = strdup(fullname.c_str());

            cout << c->nodes[col]->address << " " << c->nodes[col]->name << " " << c->
                nodes[col]->type << " " << c->nodes[col]->numFanin <<" " <<  c ->
                nodes[col] -> func <<endl;

            if (fanin > 0)
            {
                getline(inFile, s);
                istringstream iss(s);
                for (int i = 0; i < fanin; i++)
                {
                    iss >> c -> nodes[col] -> fanins[i]; // nodes drive the gate
                }
            }

        if(fanin != 0){
            c -> nodes[col] -> type = DRIVER;
            fullname = "gat_out" + add.str();
            c->nodes[col]-> outname = strdup(fullname.c_str());

            if(fanout == 0)
                c -> nodes[col] -> type = OUTPT;
        }

        col++;

        char fan[256];
        if(fanout > 1)
        {
            for(int i = 0; i < fanout; i++){
                 getline(inFile, s);
                istringstream iss(s);
                iss >> address >> fan;
                c -> nodes[col] = (NODE *) malloc (sizeof(NODE));
                c -> nodes[col] -> address = address;
                c -> nodes[col] -> name = strdup(name);
                c -> nodes[col] -> type = FROM;
                c -> nodes[col] ->numFanin = 1;
                c -> nodes[col] -> fanins = (int *) malloc (sizeof(int));
                c -> nodes[col] -> fanins[0] = address;
                c -> nodes[col] -> func = strdup(type);   /// reading type from file
                c->nodes[col]-> outname = strdup(fullname.c_str());
                cout << "Name fanout: " << c->nodes[col]-> outname << endl;

                col++;
            }
        }

        c-> numNode = col;

        cout << "Number of col: " << c-> numNode << endl << endl;
    }
    return c;
}
```

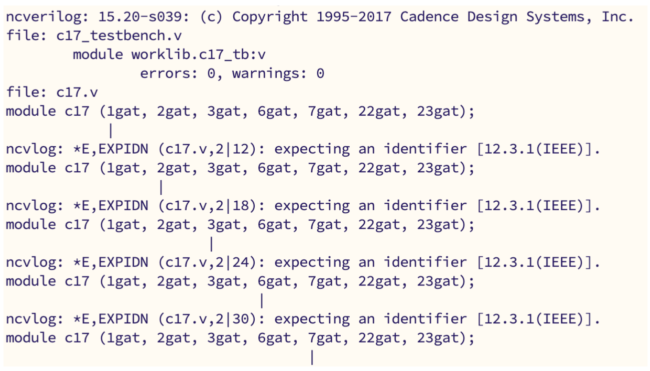
4. The hardness of this assignment / I overcome it

1. Segmentation fault when writing to a string.
   Ans: I change the string to char * in my structure NODE and CKT. Because using string only copy the string object, not the data it holds. For further information seeing Segmentation fault when using string in structure

2. To put const char* into char *, and the data lost.
   Ans: Using strdup to keep the data in the structure. The usage shown below strdup, strndup - duplicate a specific number of bytes from a string

3. Variable names have to start with an alphabetic.
   Ans: At the beginning, I only use the node name ”1gat” (the second entry on the line) to print out. However, errors like the following occur:

   Quick Reference for Verilog HDL



```
ncverilog: 15.20-s039: (c) Copyright 1995-2017 Cadence Design Systems, Inc.
file: c17_testbench.v
        module worklib.c17_tb:v
                errors: 0, warnings: 0
file: c17.v
module c17 (1gat, 2gat, 3gat, 6gat, 7gat, 22gat, 23gat);
            |
ncvlog: *E,EXPIDN (c17.v,2|12): expecting an identifier [12.3.1(IEEE)].
module c17 (1gat, 2gat, 3gat, 6gat, 7gat, 22gat, 23gat);
                  |
ncvlog: *E,EXPIDN (c17.v,2|18): expecting an identifier [12.3.1(IEEE)].
module c17 (1gat, 2gat, 3gat, 6gat, 7gat, 22gat, 23gat);
                        |
ncvlog: *E,EXPIDN (c17.v,2|24): expecting an identifier [12.3.1(IEEE)].
module c17 (1gat, 2gat, 3gat, 6gat, 7gat, 22gat, 23gat);
                              |
ncvlog: *E,EXPIDN (c17.v,2|30): expecting an identifier [12.3.1(IEEE)].
module c17 (1gat, 2gat, 3gat, 6gat, 7gat, 22gat, 23gat);
                                    |
```

5. Suggestions

No.