CAD for VLSI Design

Project Assignment 1

Benchmark Translator

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

TA: Yi-Ting Lin

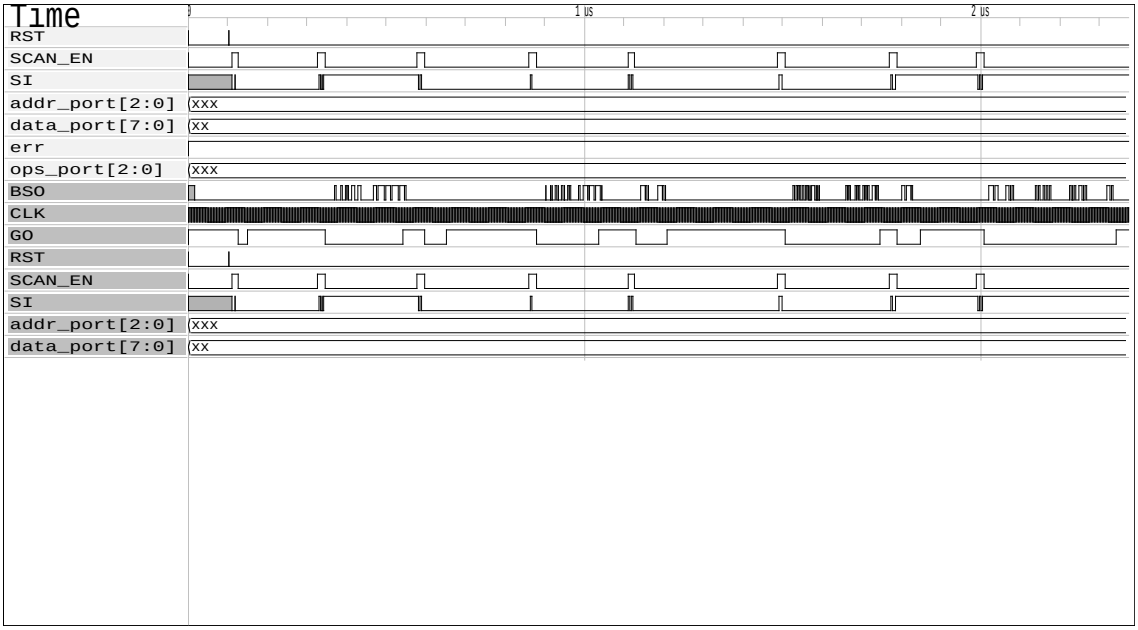
Department/Class: Electrical Engineering 4A

Name: 陳緯亭

Student ID Number: **109501201**

Contents

List Of Listings



1. How I compile and execute the program

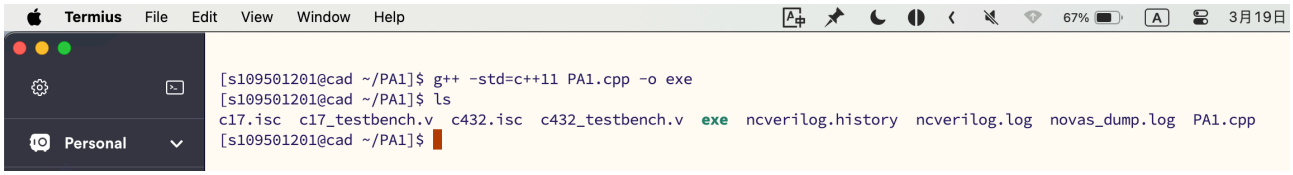


Fig 1: Generate an executable file

```
[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!!!


                 ...XXXXXXXXXXX..
               .XXXXXXX.......XXXXXX.
             XXXXX....     .......XXXX.
           .XXX.......     ....... ..XX.
          .XX.  ..             ...   .XX
         .XX..   .XXX.     .XXX.      .XX
         XXX.....XXXXX.    XXXXX     ..XX.
       ..XXX.......XXX.......XXX........XXX
    XXXXXXXX..XXXXXXXXXXXXXXXXXXXX.....XXXX.
  XXX.    XXXX..XXXXX........XXXXXX.. ...XXXXXXXXX.
 XX.      XXX.......XXXXXXXXXX.....XXXXXX      .XXX.
 .XXX.    .XX.. ..  ...................XXX         XX.
   .XXXXX. .XX.                      ..XXX        .XXX.
     ...XXXXXXXX.                  ...XXX  ..XXXXXXX.
          ..XXXXX.             ....XXXXXXXXXX...
               .XXXX..........XXXXXX.
                 .XXXXXXXXXXXXXXXXXX.
                  XXXXXXXXXXXXXXXXXX
                  XXXXXXXXXXXXXXXXXX
                  XXXXXXXXXXXXXXXXXX

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

Fig 4: Check the correctness

2. Pseudo Code

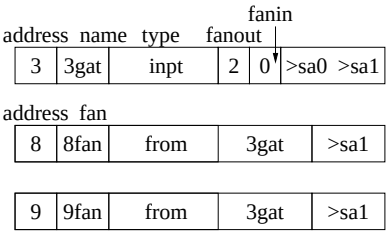


Fig 5: The Variables store the information

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 $c \rightarrow numNode$ **do**
        if $c \rightarrow nodes[i] \rightarrow type \neq INPT$ and $c \rightarrow nodes[i] \rightarrow type \neq FROM$ **then**
            \\ type and instance to $outFile$
            for $j = 0$ to $c \rightarrow nodes[i] \rightarrow numFanin$ **do**
                $k \leftarrow 0$
                while $k < c \rightarrow numNode$ **do**
                    if $c \rightarrow nodes[k] \rightarrow address == c \rightarrow nodes[i] \rightarrow fanins[j]$ **then**
                        Write ", " + $c \rightarrow nodes[k] \rightarrow outname$ 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 $col \leftarrow 0$

```
    while getline(inFile, s) do
        istringstream iss(s)
        if s[0] ==' *' then
            continue
        end if
        iss → address, name, type, fanout, fanin
        \\ Get the information from inFile and Store in c
        if fanin > 0 then
            for i = 0 to fanin do
                iss → c → nodes[col] → fanins[i]
            end for
        end if
        if fanin ≠ 0 then
            c → nodes[col] → type ← DRIVER
            c → nodes[col] → outname ← "gat_out" + address
            if fanout == 0 then
                c → nodes[col] → type ← OUTPT
            end if
        end if
        col ← col + 1
        if fanout > 1 then
            for i = 0 to fanout do
                getline(inFile, s)
                istringstream iss(s)
                iss → address, fan
                \\ Get the information from inFile and Store in c
                col ← col + 1
            end for
        end if
        c → numNode ← col
    end while
    return c
end function
```

3. PA

3.1 The degree of completion of the assignment: ALL

3.2 Code Explanation

Listing 1: Preprocessor

```
#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

```
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
};
```

```
34
35 typedef struct CKT *PCKT;
36
37
38 PCKT first_pass(ifstream& inFile, PCKT c);
39 PCKT second_pass(ifstream& inFile, PCKT c);
40 PCKT input_file(char *iscFile);
41 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

```
42 int main(int argc, char *argv[]){
43     PCKT c;
44
45     c = input_file(argv[1]);
46
47     output_file(argv[2], c);
48
49     free(c);
50
51     return 0;
52
53 }
```

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

```
54 PCKT input_file(char *iscFile){
55
56     PCKT c;
57     ifstream inFile(iscFile);
58
59     c = (PCKT) malloc (sizeof(CKT));
60
61     if(!inFile){
62         cout << "isc file can't not be open.";
63     }
64
65     c->name = iscFile;
```

```
66
67      c = first_pass(inFile, c);
68
69
70      inFile.clear(); // Clear any error flags
71      inFile.seekg(0, ios::beg); // Reset file pointer to beginning
72
73      c = second_pass(inFile, c);
74
75      cout << "NO " << c -> nodes[0] -> name << endl;
76
77      inFile.close(); // Close the file
78
79      return c;
80  }
```

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

```
81  void output_file(char *vFile, PCKT c){
82
83      ofstream outFile(vFile);
84
85      if(!outFile){
86          cout << "v can't be generate.";
87      }
88
89      outFile << "`timescale 1ns/1ps\n";
90
91      bool notFirst = 0;
92
93      // module
94      outFile << "module " << string(c->name).substr(0, string(c->name).length() - 4) <<
            " (";
95      for(int i = 0; i < c-> numNode; i++){
96          if(c -> nodes[i] -> type == INPT || c -> nodes[i] -> type == OUTPT){
97              if(notFirst)
98                  outFile << ", ";
99              outFile << c -> nodes[i] -> outname;
100             notFirst = 1;
101         }
102     }
103     outFile << ");\n";
```

```
104
105     // input
106     notFirst = 0;
107     outFile << "input ";
108     for(int i = 0; i < c->numNode; i++){
109         if(c -> nodes[i] -> type == INPT){
110             if(notFirst)
111                 outFile << ", ";
112             outFile << c -> nodes[i] -> outname;
113             notFirst = 1;
114         }
115     }
116     outFile << ";\n";
117
118     // outputs
119     notFirst = 0;
120     outFile << "output ";
121     for(int i = 0; i < c->numNode; i++){
122         if(c -> nodes[i] -> type == OUTPT){
123             if(notFirst)
124                 outFile << ", ";
125             outFile << c -> nodes[i] -> outname;
126             notFirst = 1;
127         }
128     }
129     outFile << ";\n";
130
131     // wire
132     if(c -> numDriver > 0){
133         notFirst = 0;
134         outFile << "wire ";
135         notFirst = 0;
136         for(int i = 0; i < c -> numNode; i++){
137
138             if(c -> nodes[i] -> type == DRIVER){
139                 if(notFirst)
140                     outFile << ", ";
141                 outFile << c -> nodes[i] -> outname;
142                 notFirst = 1;
143             }
144         }
145             outFile << ";\n\n";
146     }
147
148     outFile << "\n";
149
150     for(int i = 0; i < c -> numNode; i++){
151         if(c -> nodes[i] -> type != INPT
152         && c -> nodes[i] -> type != FROM){
153             outFile << c -> nodes[i] -> func << " ";
```

```
154             outFile << c -> nodes[i] -> func << c -> nodes[i] -> address << " ( ";
155             outFile << c -> nodes[i] -> outname ;
156             for(int j = 0; j < c -> nodes[i] -> numFanin; j++)
157             {
158                 int k = 0;
159                 while(k < c -> numNode){
160                     if (c->nodes[k]->address == c->nodes[i]->fanins[j]) {
161                             outFile << ", " << c->nodes[k]-> outname;
162                     }
163                     k++;
164                 }
165             }
166             outFile << " );\n";
167         }
168     }
169     outFile << "\nendmodule\n";
170 }
```

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

Listing 6: First file process

```
171 PCKT first_pass(ifstream& inFile, PCKT c){
172     string s;
173     int address, fanout, fanin;
174     string name, type;
175     int numOutput = 0, numInput = 0;
176     int maxAddress = -1;
177     int lineCount = 0;                  // column line cause by number of fanout
178 
179     while(getline(inFile, s)){
180         istringstream iss(s);
181         if(s[0] == '*') continue;
182         iss >> address >> name >> type >> fanout >> fanin;
183         lineCount += fanout;
184         maxAddress = (address > maxAddress) ? address : maxAddress;
185         if(fanin > 0) getline(inFile, s);
186         if(fanout > 1){
187             while(fanout-- > 0){
188                 getline(inFile, s);
189             }
190         }
191         if(fanout == 0) numOutput++;
192         if(fanin == 0)  numInput++;
193     }
194 
195     c -> numOutput = numOutput;
196     c -> numInput = numInput;
197     c -> nodes = (PNODE *) malloc ((maxAddress + lineCount + 8) * sizeof(PNODE));
```

```
198     cout << "LineCount: " << lineCount << endl;
199     cout << "maxAddress: " << maxAddress << endl << endl;
200 
201     return c;
202 }
```

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

```
203 PCKT second_pass(ifstream& inFile, PCKT c){
204 
205     string s;
206     int address, fanout, fanin;
207     char name[256], type[256];
208     int col = 0;
209 
210     while(getline(inFile, s)){
211         istringstream iss(s);
212         if(s[0] == '*') continue;
213         iss >> address >> name >> type >> fanout >> fanin;
214         cout << address << " " << name << " " << type << " " << fanout << " " << fanin
                 << endl;
215 
216         stringstream add;
217             add << address;
218 
219             string fullname = "gat" + add.str();
220 
221             c->nodes[col] = (NODE *)malloc(sizeof(NODE));
222             c->nodes[col]->address = address;
223             c->nodes[col]->name = strdup(name);
224             c->nodes[col]->type = INPT;
225             c->nodes[col]->numFanin = fanin;
226             c->nodes[col]->fanins = (int *)malloc(fanin * sizeof(int));
227             c->nodes[col]->func = strdup(type);
228             if(strcmp(type, "buff") == 0)
229                 c->nodes[col]->func = strdup("buf");
230             c->nodes[col]-> outname = strdup(fullname.c_str());
231 
232             cout << c->nodes[col]->address << " " << c->nodes[col]->name << " " << c->
                    nodes[col]->type << " " << c->nodes[col]->numFanin <<" " <<  c ->
                    nodes[col] -> func <<endl;
233 
234             if (fanin > 0)
235             {
236                 getline(inFile, s);
```

```
237                 istringstream iss(s);
238                 for (int i = 0; i < fanin; i++)
239                 {
240                     iss >> c -> nodes[col] -> fanins[i]; // nodes drive the gate
241                 }
242             }
243 
244         if(fanin != 0){
245             c -> nodes[col] -> type = DRIVER;
246             fullname = "gat_out" + add.str();
247             c->nodes[col]-> outname = strdup(fullname.c_str());
248 
249             if(fanout == 0)
250                 c -> nodes[col] -> type = OUTPT;
251         }
252 
253         col++;
254 
255         char fan[256];
256         if(fanout > 1)
257         {
258             for(int i = 0; i < fanout; i++){
259                  getline(inFile, s);
260                 istringstream iss(s);
261                 iss >> address >> fan;
262                 c -> nodes[col] = (NODE *) malloc (sizeof(NODE));
263                 c -> nodes[col] -> address = address;
264                 c -> nodes[col] -> name = strdup(name);
265                 c -> nodes[col] -> type = FROM;
266                 c -> nodes[col] ->numFanin = 1;
267                 c -> nodes[col] -> fanins = (int *) malloc (sizeof(int));
268                 c -> nodes[col] -> fanins[0] = address;
269                 c -> nodes[col] -> func = strdup(type);   /// reading type from file
270                 c->nodes[col]-> outname = strdup(fullname.c_str());
271                 cout << "Name fanout: " << c->nodes[col]-> outname << endl;
272 
273                 col++;
274             }
275         }
276 
277         c-> numNode = col;
278 
279         cout << "Number of col: " << c-> numNode << endl << endl;
280     }
281     return c;
282 }
```

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.