EE1003 Introduction to Computer I

Final Project: Design Synthesis

Due Date: 2022/1/18 (Tue.) 23:59:59

Background

Design procedures or methodologies specify hardware that will implement a desired behavior. The design effort for small circuits may be manual, but the industry relies on **automated synthesis tools** for designing massive integrated circuits. Although it can meet our needs quickly and conveniently, but the cost of these Electronic Design Automation (EDA) tools license is too expensive. Therefore, many companies are now committed to the development of their own synthesis tools to reduce the cost.

Accordingly, NCU wants to develop their own synthesis tool that can automatically generate gate-level circuits from design descriptions. Since this is a massive project, NCU has separated the whole project into different small portions. One team has accomplished the transformation from design description to a state table, and the other team has implemented an algorithm to minimize complex logical expressions by Karnaugh map (K-map) minimization method. Your mission is to integrate these teams' programs and complete the synthesis tool which can automatically produce the corresponding circuit design based on the given design descriptions.

Problem Description

The flow diagram of the NCU synthesis tool is shown in Fig 1. In the beginning, according to the specification, we can obtain a related state diagram. In the next step, we can get the binary-coded state table by obtaining binary values from the state diagram. Then, the user needs to choose the type of flip-flops he/she are going to use. Next, the tool will derive the minimized equations of input and output via Karnaugh map minimization method. Finally, depending on the minimized result, we can obtain the desired circuit design in a specific text format. As you can see from the figure, the red braces identify the unfinished portion of the tool. You need to finish all the unfinished stages and complete the synthesis tool.

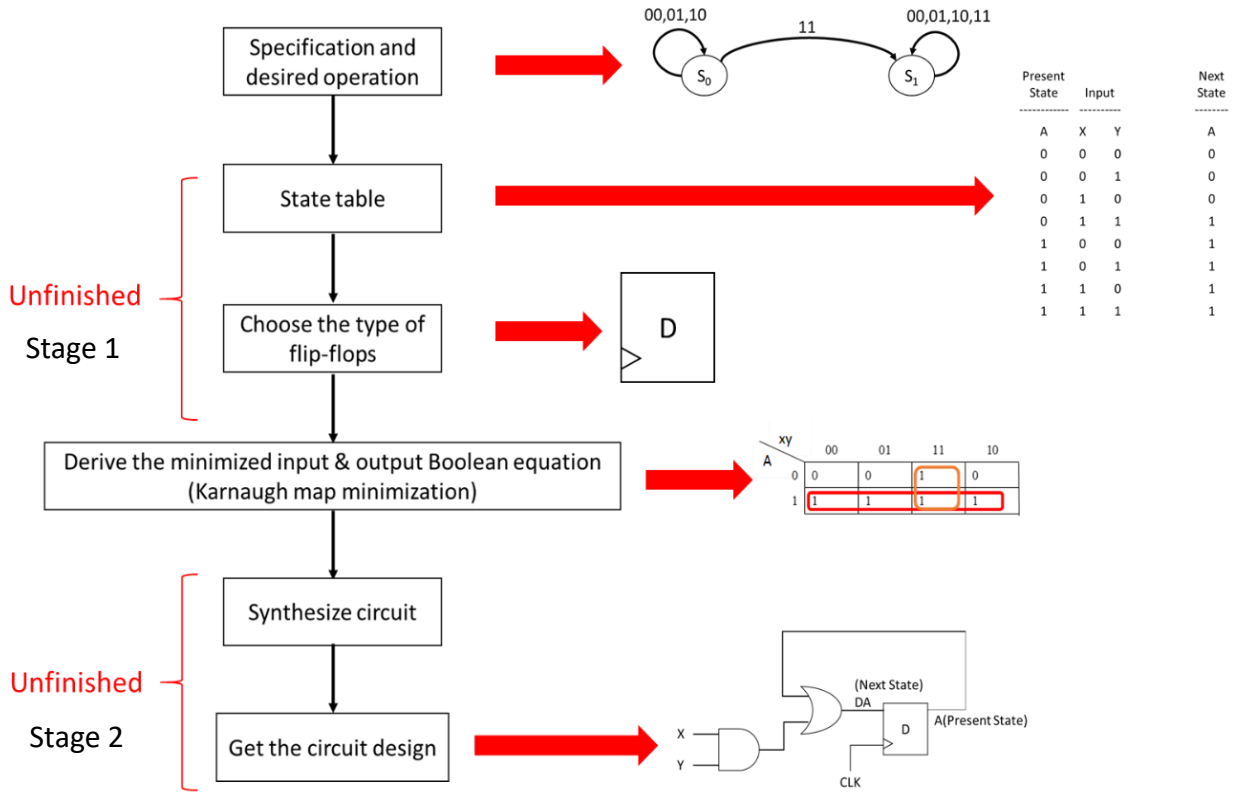


Fig. 1. NCU synthesis tool flow diagram

Fig. 1 shows the flow diagram of design procedure.

Stage 1:

Input:

As shown in Fig. 1, the input of stage 1 is a state table. An example of a state table is shown in Fig. 2. To easily read this table into your program, the state table will be given in the input file named "state_table.txt", which includes the information of Fig. 2. Note that we assume all Flip-flops are DFF to simplify the problem.

Present State	Input		Next State
A	X	Y	A
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Fig. 2. State table

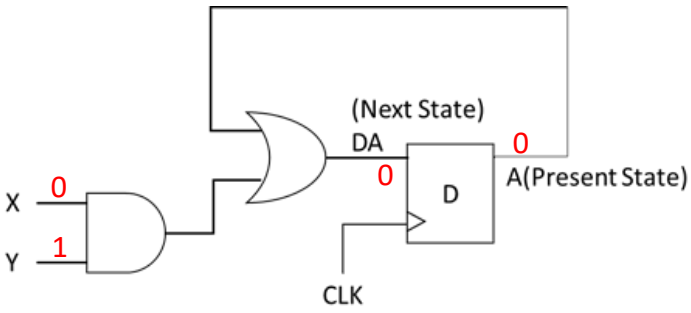


Fig. 3. Circuit design

An example is shown follow:

```
Flip-Flop: D

//Present(A)
0
0
0
0
1
1
1
1

//Input(x,y)
0 0
0 1
1 0
1 1
0 0
0 1
1 0
1 1

//Next(A)
0
0
0
1
1
1
1
1

//Output(0)
```

Note that if there is a “0” in the parentheses, it means that the circuit doesn’t exist the corresponding node. In the example above, it shows this circuit has no output

because there is a “0” in the output bracket.

Output:

Your program should take the input file and generate a corresponding Karnaugh map stored in the file named ”kmap_in.txt” which is used for the Karnaugh map minimization program developed by the other teams. An output file example is shown as follows. The details will be described in the next section.

```
1
3
0 0 1 0 1 1 1 1
```

Karnaugh Map Program User Guide:

The first line in the input file is an integer indicates the number of Boolean functions. Then each Karnaugh map will be represented by two lines. The first line is an integer to show the number of variables in the Boolean function. The second line represents the Boolean function in the Karnaugh map format. In the second line, the integers are corresponding to the value put in the squares of the Karnaugh map from top to down and left to right order. For each block in the Karnaugh map, the value can be either 1 for logic 1 and 0 for logic 0.

Following is the example of the input file where the comments is just for your reference and will not be appear in the file:

```
1       //number of Boolean functions
3       //number of Boolean functions variables
0 0 1 0 1 1 1 1
```

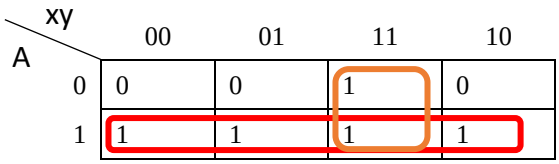
The K-map minimization program will then perform logic reduction based on the rule mentioned in logic design class.

In this example, the K-map minimization program will perform the reduction as follows:

k=3

A \ xy	00	01	11	10
0	0	0	1	0
1	1	1	1	1

Sequence order: 0 0 1 0 1 1 1 1



Therefore, the user can obtain the minimized Boolean function (SOP form) as "xy + A". Finally, K-map minimization program will generate an output file named "kmap_out.txt" and it represents the SOP form in the first 3 lines of the output file as follows where the first line gives the number label of the Boolean function,

```
#1        //The first function with the order (A, x, y)
2 1 1     // The term xy
1 2 2     // The term A
```

The value in the Boolean function can be 0, 1, or 2 (don't care).

To integrate this program into your code, you need to include their source code named "kmap.cpp", which we will provide to you, and use the function called ***void top()*** in your code. This function will automatically produce and store the minimized result in the file named "kmap_out.txt".

Stage 2:

Input:

After previous step, according to the Fig. 1, your program should take the result file named "kmap_out.txt" and process it. An input file example is shown follow:

```
#1
2 1 1
1 2 2
```

Output:

After taking the input file, your program should produce a corresponding circuit, which is shown in Fig. 3. Notice that NCU has requested that this synthesis tool has to describe the circuit in **benchmark** format. An example is shown follow:

```
# 2 inputs
# 0 output
# 0 inverter
# 1 flip-flop
# 2 gates (1 AND + 1 OR)

INPUT(x)
INPUT(y)

N1 = AND(x,y)
DA = OR(A,N1)
A = D(DA,CLK)
#END
```

The first 5 lines starting with # are the comment lines. These comment lines provide basic information of the benchmark. Then a blank line appears. After that, all the input ports are specified one per line. Then a blank line appears again, follows by all the output ports. A blank line appears again, follows by circuit connection information. All interconnect wires in the circuit are named as "Nx" (x: 1~n) and all the output nodes of inverter are named as "!m" (m: the input node's name of the inverter). Notice that if the circuit doesn't exist input or output port, you don't have to print that port.

We will verify your program by comparing the circuit produced by your program with our solution.

Submission Requirement

You have to submit a source code file (not the entire project) named as StudID_Final.cpp (ex: 986253465_Final.cpp) and a report named StudID_Name_Final_report.pdf (ex: 986253465_陳聿廣_Final_report.pdf). Please upload them to ee-class. Note that the only acceptable report file format is .pdf, no .doc/.docx or other files are acceptable. **BE SURE to follow the naming rule mentioned above. Otherwise, your program will be not graded.**

We will use C++ 11 standard and will use the workstation to judge your program by the following commend:

$program.out input.txt result.txt

where the first term is the executable file, the second term is the input file name, and

the third term is the output file name. A workstation user manual will be released soon.

We don't restrict the report format and length. In your report, you have to at least describe:

1. How to compile and execute your program; (You can use screenshot to explain)
2. The completion of the assignment; (If you complete all requirements, just specify all)
3. The hardness of this assignment and how you overcome it;
4. Any suggestions about this programming assignment?

You can also put anything related to the Final Project in your report, such as pseudocode, control flow diagram, programming developing thought, etc.

Your program will be judged with Code::Blocks 20.03 and GNU GCC Compiler (MinGW-W64 project version 8.1.0, 32/64 bit, SHE). We will use C++ 11 standard.

Grading

The grading is as follows:

(1) Correctness of your code: 70%

(2) Readability of your code: 10%

(3) The report: 20%

NO demo session will be held for final project!!!!

NO LATE SUBMISSION is accepted!!!!! Please submit your assignment on time.

If you have questions, please E-mail to both me (andyygchen@ee.ncu.edu.tw) and TA 黃柏燁&金昌明 (alec2515@gmail.com & abcd29417557@gmail.com).

Reference

[1] K-map Logic Minimization: https://www.allaboutcircuits.com/technical-articles/karnaugh-map-boolean-algebraic-simplification-technique/

[2] Digital Logic class OCW: https://youtu.be/Fx_BNWSAApQ