Logic Design Lab 1: Fibonacci Number Detector Report

110062219

March 30, 2022

1 Design

We are to carry out a 4-bit Fibonacci Number Detector in the Lab 1. It might be the most intuitive approach to adopt the **Behavaior Description**, in which we first enumerate all Fibonacci number less than 16 — 0, 1, 2, 3, 5, 8, 13 — and use `always` and `case` blocks to detect them.

1.1 Karnaugh Map

Since we are interested in hardware, we could take a further and more deep look at the logic and Boolean algebra.

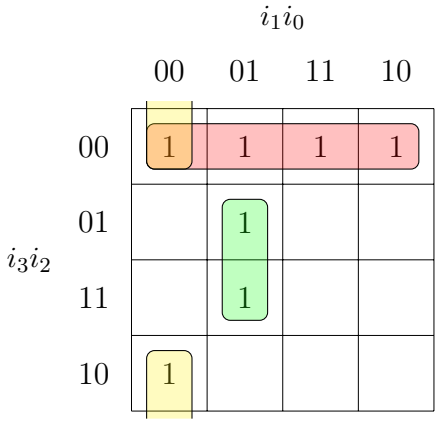

Figure 1: Karnaugh Map

The 4-bit integer input could be consider to be a 4 bits array $i_3i_2i_1i_0$. Then our detector f could be write as sum of minterms $\sum m(0, 1, 2, 3, 5, 8, 13)$. By the means of *Karnaugh Map* (Figure 1), we could select the implicants (red for $i_3'i_2'$, green for $i_2i_1'i_0$ and yellow for $i_2'i_1'i_0'$) and simplify

$$f = i_3'i_2' + i_2i_1'i_0 + i_2'i_1'i_0'$$

. As a consequence, we could have the module come in **Dataflow Description** method.

1.2 Circuit Diagram

If we would like to make our module more concrete, then we could draw the circuit diagram like Figure 2 based on the K-map and implement the module by **Gateway-level Description**.

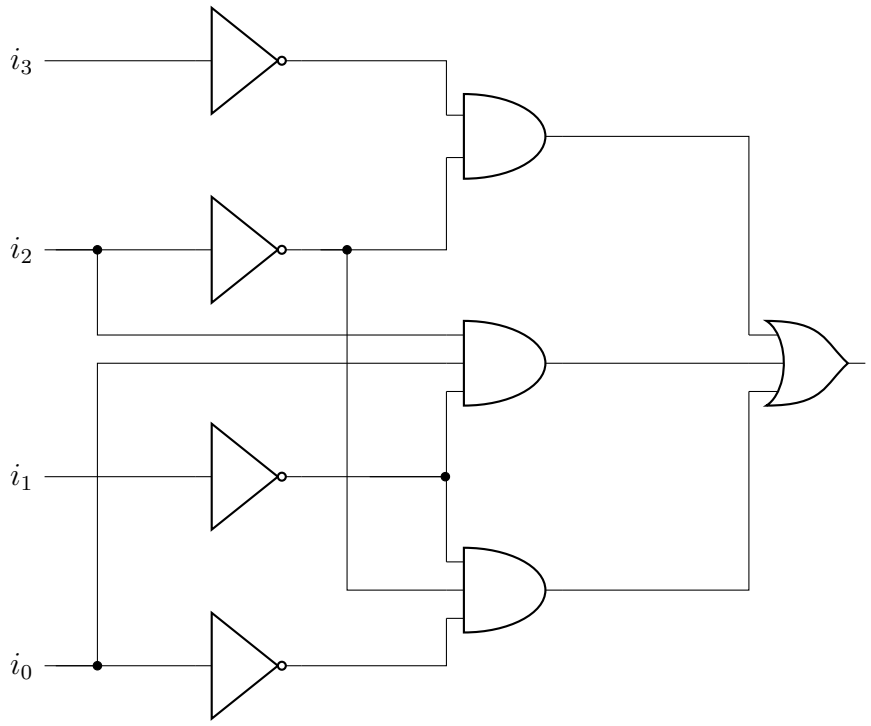

Figure 2: Circuit Diagram

1.3 Result

In addition to the test bench module alone, I wrote a helper module which take i and the outputs of each type of detectors as input and determine whether the output is either unknown or high impendence, or the output is conflict with the fact so that the condition we check is more clear.

Figure 3 shows the result of `ncverilog`.

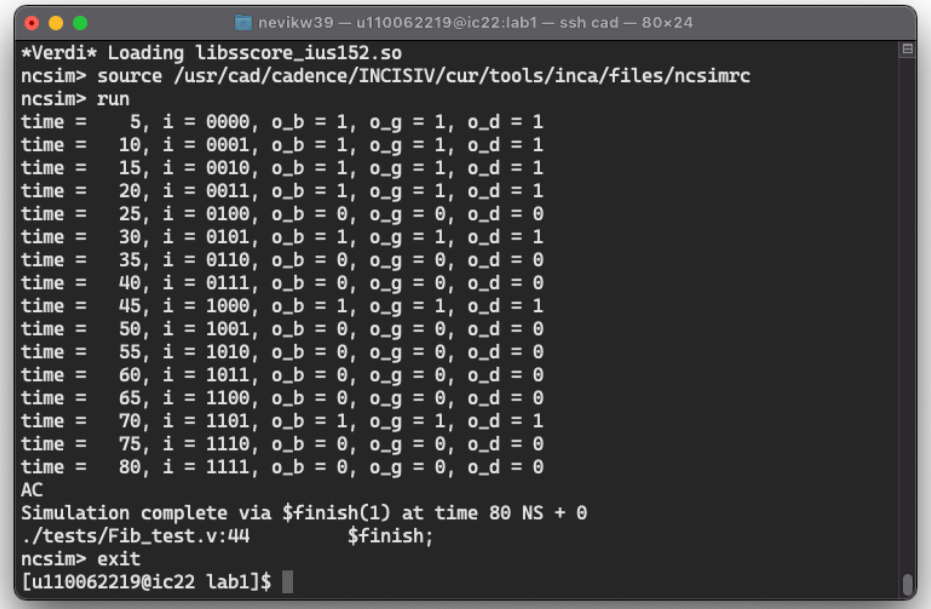

Figure 3: Screenshot

2 Problems

Since it was my very first time to write *Verilog*, it's inevitable and natural that I encountered several problems.

In the beginning, I ran into the problem that I wanted to use the remote SSH feature of VS Code, but `ncverilog` is at the node inside the CAD server. After searching some resources, I found out how to configure so that I could connect to the node via CAD server as a proxy jump server.

Moreover, it seemed that some environment variables weren't set properly when connecting through VS Code. I tried to override the `$PATH` manually

and switch to `bash`, and it worked sometimes, but `ncverilog` got license failed once in a while. After searching some resources, I found out it's because VS Code use `bash` as default and not run `.profile` properly, so I returned to default shell `tcsh` and select it as the default terminal of VS Code.

Last but not least, I had some difficult time to draw the circuit diagram (Figure 2) in LaTeX, though to draw the K-map was much more easy.

3 Reflection

I really learnt a lot in this Lab. To begin with, I wrote my debut *Verilog* codes and learnt its fundamental again in a more comprehensive and practical view.

Furthermore, I acquired more knowledge about SSH and Linux shell. Although it's hard to draw the circuit diagram in LaTeX, I'm quite satisfied with my result, which gives me a sense of accomplishment.

In the end, since there's no solid and definite specification on the report, so I'm not sure if my format is acceptable.