

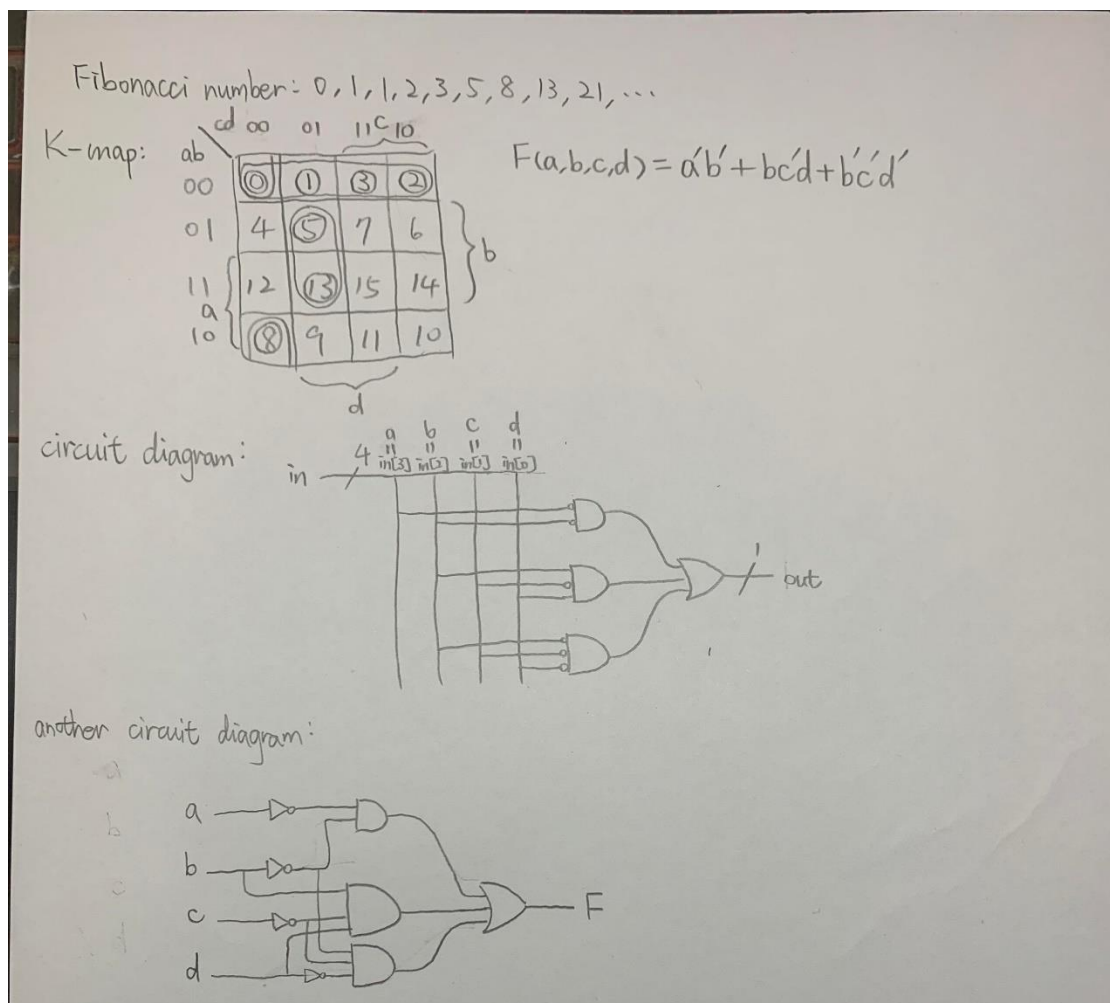
## Design Process

109021115 吳嘉濬

由命題得知我們要設計的是 4-bit 的 Fibonacci number detector，即從 0 到 15 ( $= 2^4 - 1$ ) 之間的費式數偵測器，所以我先找出在這個區間內所有的費式數，即 0,1,2,3,5,8,13 (對應到 Behavior Description 的方法)；隨後使用 K-map 的方法找出最佳的 SOP 解 (對應到 Dataflow Description 的方法)；緊接著利用剛剛得到的結果繪製出相對應的 circuit diagram，以便之後再寫 Gate Level Description 方式的 code 時增加閱覽的便利性，以提升 coding 的效率。

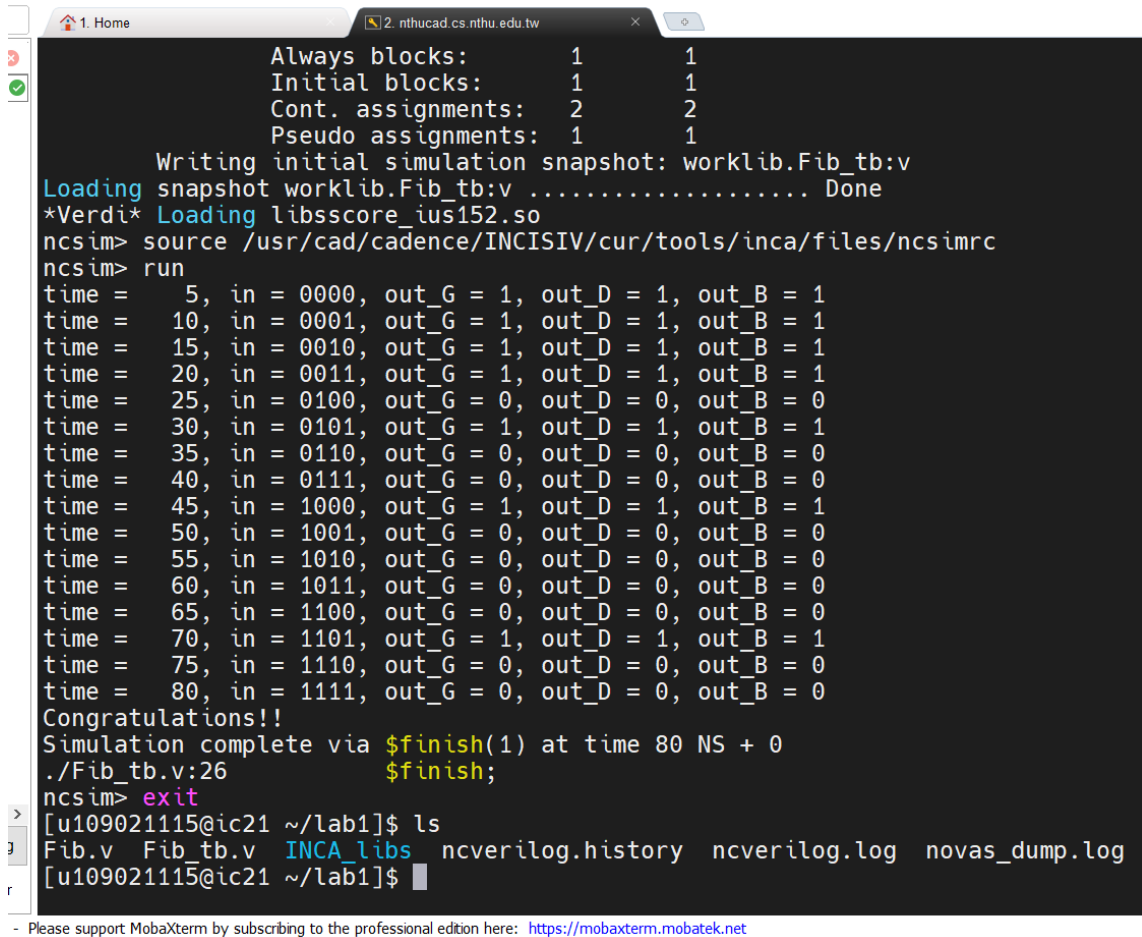
紙本計算的部分結束後，正式進入 CAD server 實際把三種方法的 module 給寫出來，再把測試用的 testbench 也寫好，最後，執行完成的 code，看是否有 bug 出現；在一波 debug 環節過後，成功完成設計。

## Circuit Diagram & K-map



## Execution Result

雖然中間有遇到一些小小 bug，但最終還是很順利的執行成功，見下圖：



```
1. Home 2. nthucad.cs.nthu.edu.tw
Always blocks:      1      1
Initial blocks:     1      1
Cont. assignments:  2      2
Pseudo assignments: 1      1
Writing initial simulation snapshot: worklib.Fib_tb.v
Loading snapshot worklib.Fib_tb.v ..... Done
*Verdi* Loading libsscore_ius152.so
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
time = 5, in = 0000, out_G = 1, out_D = 1, out_B = 1
time = 10, in = 0001, out_G = 1, out_D = 1, out_B = 1
time = 15, in = 0010, out_G = 1, out_D = 1, out_B = 1
time = 20, in = 0011, out_G = 1, out_D = 1, out_B = 1
time = 25, in = 0100, out_G = 0, out_D = 0, out_B = 0
time = 30, in = 0101, out_G = 1, out_D = 1, out_B = 1
time = 35, in = 0110, out_G = 0, out_D = 0, out_B = 0
time = 40, in = 0111, out_G = 0, out_D = 0, out_B = 0
time = 45, in = 1000, out_G = 1, out_D = 1, out_B = 1
time = 50, in = 1001, out_G = 0, out_D = 0, out_B = 0
time = 55, in = 1010, out_G = 0, out_D = 0, out_B = 0
time = 60, in = 1011, out_G = 0, out_D = 0, out_B = 0
time = 65, in = 1100, out_G = 0, out_D = 0, out_B = 0
time = 70, in = 1101, out_G = 1, out_D = 1, out_B = 1
time = 75, in = 1110, out_G = 0, out_D = 0, out_B = 0
time = 80, in = 1111, out_G = 0, out_D = 0, out_B = 0
Congratulations!!
Simulation complete via $finish(1) at time 80 NS + 0
./Fib_tb.v:26      $finish;
ncsim> exit
[u109021115@ic21 ~/lab1]$ ls
Fib.v  Fib_tb.v  INCA_libs  ncverilog.history  ncverilog.log  novas_dump.log
[u109021115@ic21 ~/lab1]$
```

- Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

## The problem I faced and how I deal with it.

第一次在執行 code 時發現 error 高達 10 個，原本以為我是不是犯了甚麼很嚴重的錯誤，仔細一看才發現原來是我把 1'b0 的””“全部都打成”`“了，還好只是簡單的語法錯誤，還以為我那天不用睡覺了。

另一個遇到的問題是：我一開始把 Fib.v 寫完後，回想了一下我以前是怎麼存檔退出的，雖然不是很確定，但依稀感覺是 ctrl + z，我沒有多想就這樣按了下去，結果當我想要再次打開檔案時，出現了以下畫面：

```

E325: ATTENTION
Found a swap file by the name ".Fib.v.swp"
    owned by: u109021115    dated: Wed Mar 30 23:33:17 2022
    file name: ~u109021115/Fib.v
    modified: YES
    user name: u109021115    host name: ic21
    process ID: 62345 (still running)
While opening file "Fib.v"
    dated: Thu Mar 31 00:40:25 2022
    NEWER than swap file!

(1) Another program may be editing the same file.  If this is the case,
    be careful not to end up with two different instances of the same
    file when making changes.  Quit, or continue with caution.
(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r Fib.v"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file ".Fib.v.swp"
    to avoid this message.

Swap file ".Fib.v.swp" already exists!
[O]pen Read-Only, (E)dit anyway, (R)ecover, (Q)uit, (A)bort:

```

我後來推測出這是因為我使用 `ctrl + z` 不正常的方式離開編輯頁面，沒有正常退出所致，如果按 `E`，`edit anyway` 的話，系統會給你一個全部空白的檔案，當下我看到時心臟差點承受不住。還好當我重新進到這個畫面時，如果按 `R`，`recover` 的話，可以找回原本編輯的檔案，但即使如此，往後要開這個檔案時，都會出現這個頁面，唯有 `rm Fib.v` 再重新 `vim Fib.v` 才可以不用再看到這個頁面。

從此之後，我永遠記得要用 `:wq` 存檔並離開檔案。

## What have I learned from this lab?

在 CAD server 底下實作時，雖然不是第一次在 linux 系統下做事，但因為有很長一段時間沒接觸，所以對一些指令還是挺生疏的。藉由這次 lab 的練習，總算是讓我找回了在 linux 系統下操作的感覺。

另外，這是我第一次用 verilog 硬體語言去設計東西，感覺還蠻新鮮的。和 C 語言相比，verilog 其實複雜許多，語法也不盡相同，像是程式中出現的數字都要標明是用多少 bit、哪種資料型態下的數字：如 `1'b0`, `4'd10`；還有，在 verilog 底下沒有 `i++` 的語法，只有 `i = i + 4'd1` 的寫法，且只能出現在 testbench 裡。

經由這次機會，除了了解基本的 verilog 語法外，我也算是深刻的感受出它和 C 兩者之間的差異。看得出來 verilog 是個邏輯清楚且嚴謹的語言，即便這樣，一旦想起它是專門用來設計硬體物件時，便會覺得會如此嚴謹也是合情合理的。

最後，在這次 lab 中還有學到一樣非常重要的事情——不要輕易嘗試 `:wq` 以

外的方式關閉檔案。