Lab 1 Fibonacci number detector

105062139_林楷宸

條件要求：

題目：Fibonacci number

範圍： 0 ~ 15

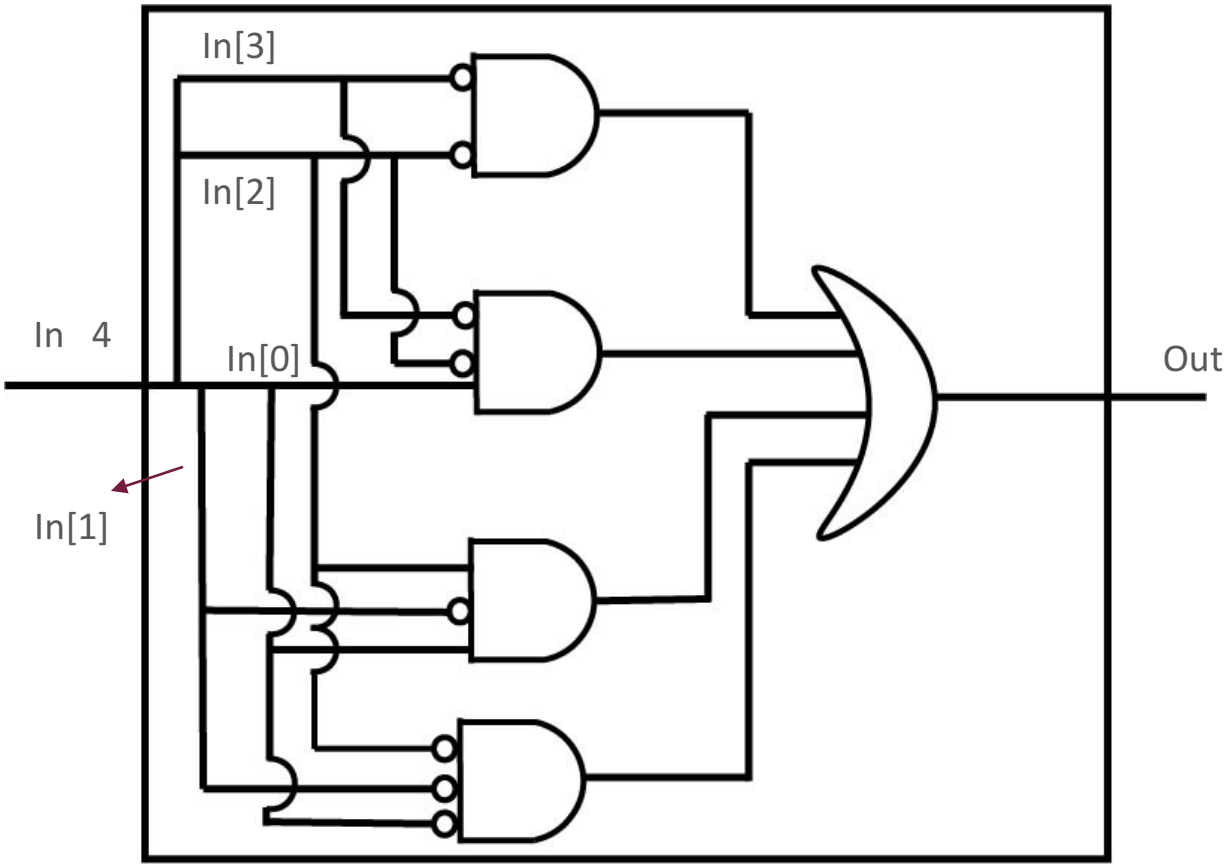
K - map：

ab \cd	**00**	**01**	**11**	**10**
00	1	1	1	1
01	0	1	0	0
11	0	1	0	0
10	1	0	0	0

K-map 結論:

~a ~b +
~a ~c d +
b ~c d +
~b ~c ~d

Schematic :



PS

a = in[3]　　b = in[2]

c = in[1]　　d = in[0]

模擬結果：

```
Loading snapshot worklib.fib_tb:v ..................... Done
*Verdi3* Loading libsscore_ius141.so
*Verdi3* : Enable Parallel Dumping.
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
time =  5, in = 0000, outG = 1, outD = 1, outB = 1
time = 10, in = 0001, outG = 1, outD = 1, outB = 1
time = 15, in = 0010, outG = 1, outD = 1, outB = 1
time = 20, in = 0011, outG = 1, outD = 1, outB = 1
time = 25, in = 0100, outG = 0, outD = 0, outB = 0
time = 30, in = 0101, outG = 1, outD = 1, outB = 1
time = 35, in = 0110, outG = 0, outD = 0, outB = 0
time = 40, in = 0111, outG = 0, outD = 0, outB = 0
time = 45, in = 1000, outG = 1, outD = 1, outB = 1
time = 50, in = 1001, outG = 0, outD = 0, outB = 0
time = 55, in = 1010, outG = 0, outD = 0, outB = 0
time = 60, in = 1011, outG = 0, outD = 0, outB = 0
time = 65, in = 1100, outG = 0, outD = 0, outB = 0
time = 70, in = 1101, outG = 1, outD = 1, outB = 1
time = 75, in = 1110, outG = 0, outD = 0, outB = 0
time = 80, in = 1111, outG = 0, outD = 0, outB = 0
Simulation complete via $finish(1) at time 80 NS + 0
./fib_tb.v:37   $finish;
ncsim> exit
[dld10054@ic27 ~/lab1]$
```

問題討論：

verilog 的語法跟上學期所學的 c 語言差很大，上學期所學的語法邏輯，想法的實踐無法套用在 verilog 裡。就好像之前問過學長為何麼大家都說邏設不好學，他給我的答案是這語言不好上手。一開始我以為他的意思只是有很多新東西要背。但實際接觸過後才發現，不只是有新東西，更重要的是許多語法要去釐清。就好像 reg, wire, assign，這三個定義，還有 integer, parameter ，一開始都完全分不清楚到底什麼時候要用什麼。只能懞懞懂懂的照抄。在近一步上網查資料後才知道。

1 以 integer 和 parameter 來說，一開始一直搞不懂為什麼，delay 要用 parameter 宣告，但是 i 卻是用 integer 來宣告。上網查資料後才知道，parameter 的功能 有點類似於 c 語言 的 define 宣告一次後，整個程式碼就不能再改他的值了，而 integer 就像 c 語言的 int 宣告。 是可以在程式中更改他的值。

2 reg 和 wire 及 assign 了話，老實說，看了很多資料，似乎懂了一些，但缺少了實戰的經驗，有許多地方還處於一個模糊的概念上。但大概可以得到一個結論是，wire 基本上 會代表 兩個 gate 之間的橋樑，只是一條電路，不具儲存功能，無驅動能力，而 reg 了話，具有儲存功能，有驅動的能力，有時可以取代 wire。以及 wire 使用時必要搭配 assign。網路上也有說：大體上來說，wire 和 reg 都類似於 C/C++的變數，但若此變數要放在 begin...end 內，該變數就須使用 reg，在 begin...end 之外，則使用 wire。

（引用 http://www.cnblogs.com/oomusou/archive/2007/10/10/9193

39.html）還有一個小結論是以輸入訊號來分，

輸入信號	不知道上一級是暫存器輸出還是組合邏輯輸出	輸出信號則由你自己來決定是組合邏輯輸出還是暫存器輸出
結論	對於本級來說就是一根導線，也就是 wire 型	wire 型、reg 型都可以
一般來說	整個程式的設計之外部輸出(即最頂層 module 的輸出)，要求是暫存器輸出，用 reg	

3 這次寫 verilog 也嘗試了許多另類的語法嘗試，像是 for loop ，在 c 語言中，常常把許多東西直接放在小括號內，i=i+1 也常常也成 i++，這次在寫 verilog 時也想嘗試了一下，把原來

```
for(i=0; i<16; i = i + 1)
        begin
                #(5)
                $display("time = %2d, in = %b, outG = %b, outD = %b, outB = %b",$time, in, outG, outD, outB);
                in = in + 4'b0001;
        end
        $finish;
```

改成

```
for(i=0; i<16; i++, in = in + 4’ b0001)
        begin
                #(5)
                $display("time = %2d, in = %b, outG = %b, outD = %b, outB = %b",$time, in, outG, outD, outB);
        end
        $finish;
```

才知道 verilog 的語法並不能這麼寫，一定要分開寫才行。

參考資料：

integer, parameter : http://ppt.cc/WglBS

assign, wire, reg:

http://frankchang0125.blogspot.tw/2009/05/wire-vs-reg.html

https://www.ptt.cc/bbs/Electronics/M.1303030120.A.0A1.html