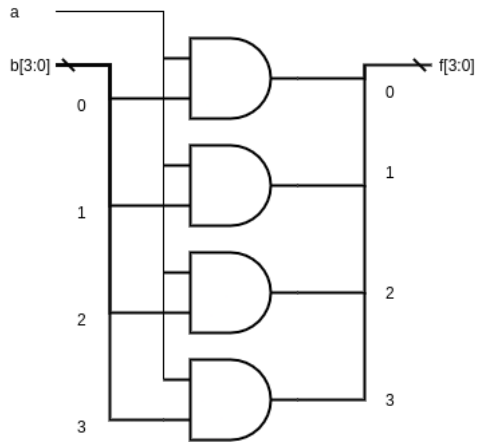
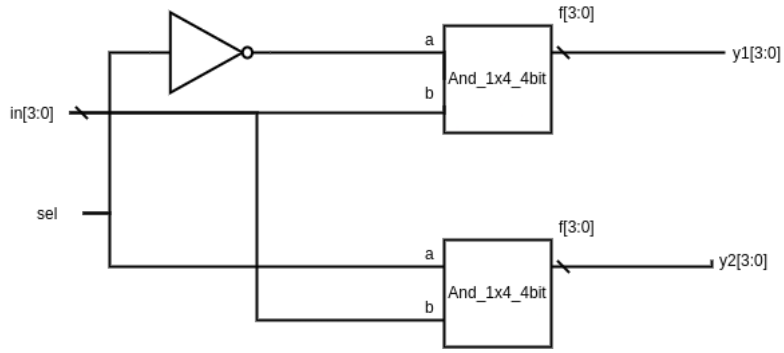
20220924 Lab 1 Gate-Level Verilog

Dmux 1x4_4bit:

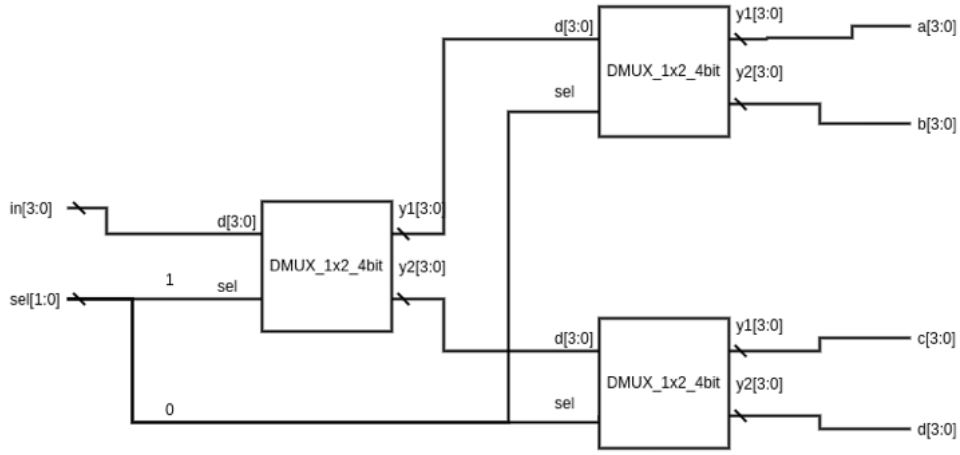
Logic Diagram:



(And_1x4_4bit)

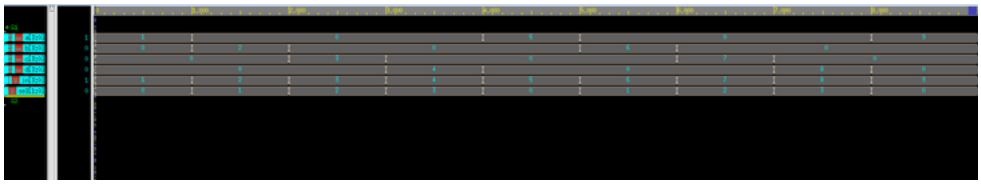


(DMUX_1x2_4bit)



(DMUX_1x4_4bit)

Wave Diagram:



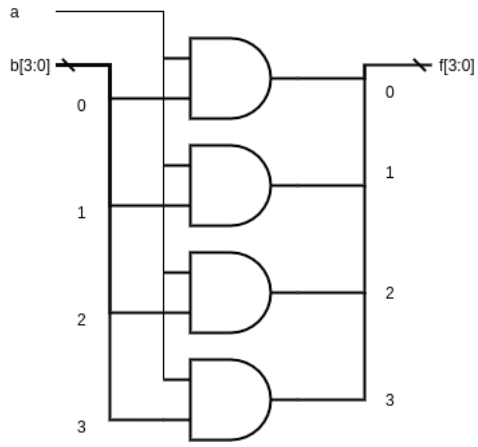
Report

透過以上submodule，包含：將4個bit與一個bit進行bitwise and，1x2的DMUX，我們能組合出1x4的DMUX如上圖。

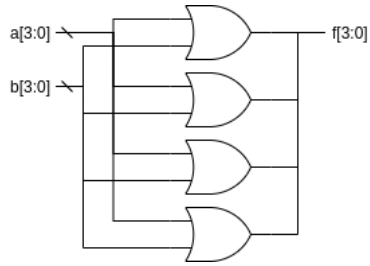
利用sel[1]決定輸出會在a, b這組還是c, d這組，再用sel[0]來決定最終輸出是在該組中的哪一個位置。如果將輸出位置以0-index編號，sel[i]相當於是決定輸出位置編號的第i個bit的值。
此外，觀察將DMUX_1x2組合成DMUX_1x4的過程，我們可以依樣畫葫蘆，在DMUX_1x4的每個輸出後再接上一個DMUX_1x2，即可組合出DMUX_1x8。
依此類推，我們可以使用DMUX_1x2組合出任意1x2^k 的DMUX，並且由於二元樹的性質，我們可以計算組合出1x2^k 的DMUX需要$2^k - 1$個DMUX_1x2。
在testbench的部份，我們在in[3:0]非零時枚舉sel[1:0]的所有可能，並確認輸出的位置是正確的。

CrossBar_2x2_4bit:

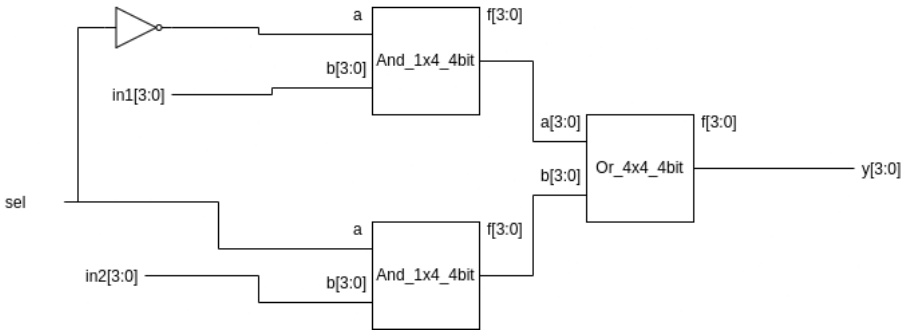
Logic Diagram:



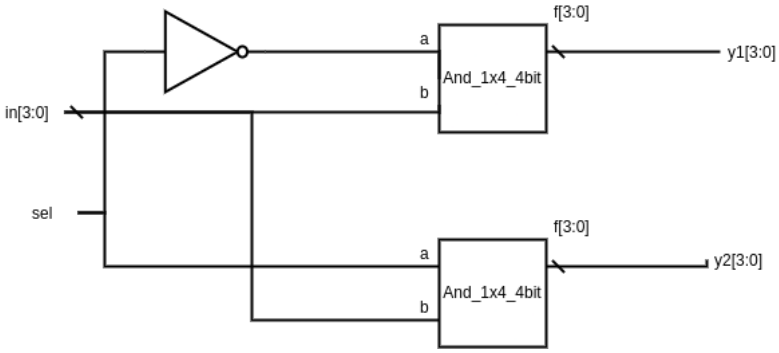
(And_1x4_4bit)



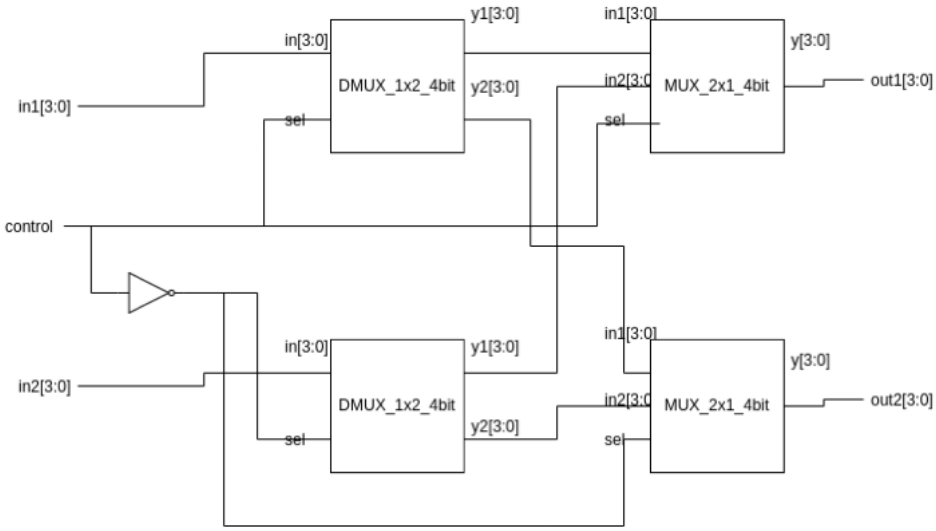
(Or_4x4_4bit)



(MUX_2x1_4bit)

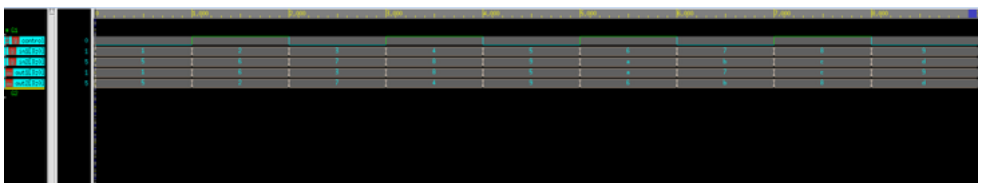


(DMUX_1x2_4bit)



(Crossbar_2x2_4bit)

Wave Diagram:



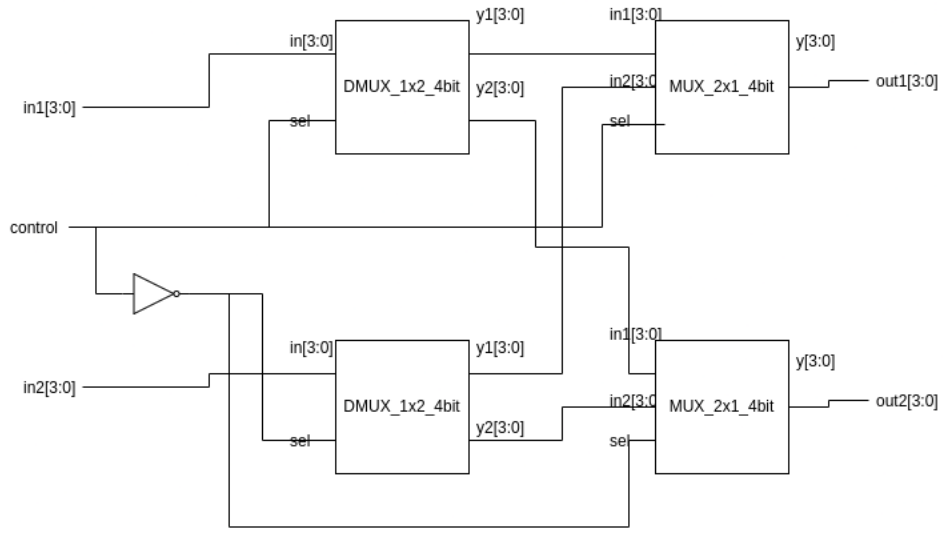
Report

利用DMUX與MUX將輸入交換位置。

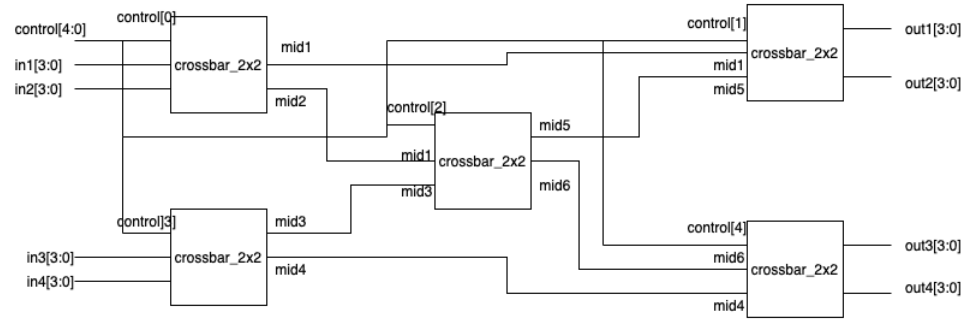
我其實不太懂為什麼CrossBar需要2個DMUX跟2個MUX，因為似乎只需要兩個DMUX跟兩個bitwise OR gate、或兩個MUX，就能作出一樣的行為。我的理解是這樣的結構比較易於推廣到更多輸入與輸出的版本。
testbench的部份，由於control只有兩種可能，對於不同的in1，in2枚舉過control的值後，應該就替正確性提供了不小的保證。

CrossBar_4x4_4bit:

Logic Diagram:

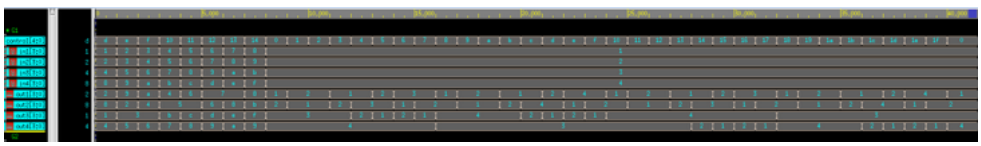


(CrossBar_2x2_4bit)



(CrossBar_4x4_4bit)

Wave Diagram:

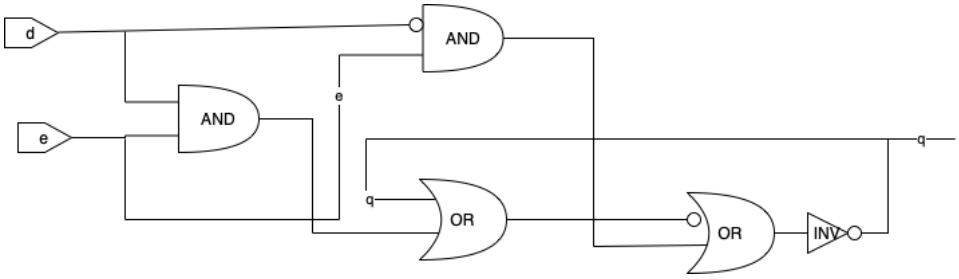


Report:

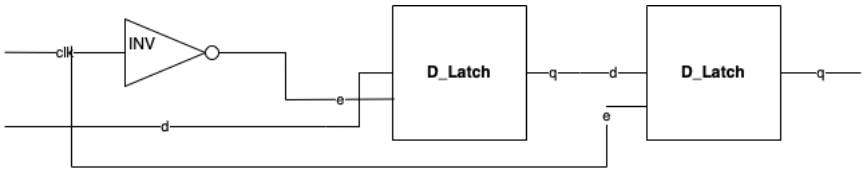
利用五個2x2_Crossbar以及一個5bits的control來交換in1-in4對應到out1-out4的位置。
Crossbar_4x4的測試是枚舉所有control的可能，來看in對應到out的位置對不對。
而我們可以發現(out1, out2, out3, out4)在所有的control情況中並沒有出現(in3, in4, in1, in2), (in3, in4, in2, in1), (in4, in3, in1, in2), (in4, in3, in2, in1)的可能，所以這四個情況是不會出現在Crossbar_4x4的output的。

Toggle Flip Flop:

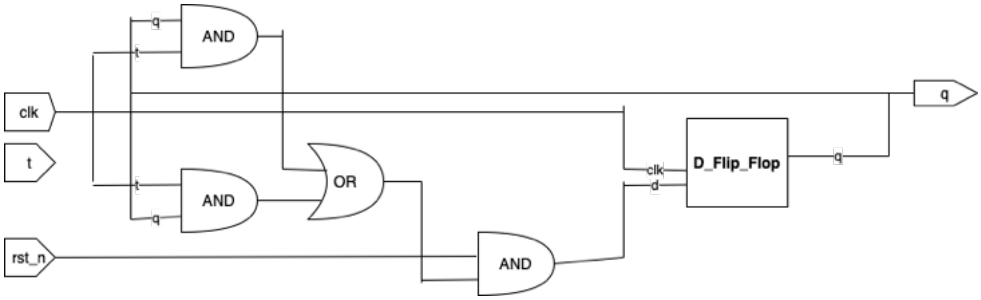
Logic Diagram:



(D_Latch)

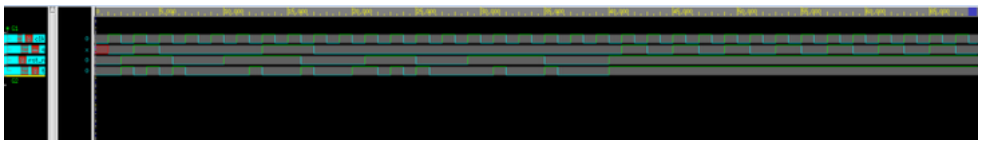


(D_Flip_Flop)



(Toggle_Flip_Flop)

Wave Diagram:

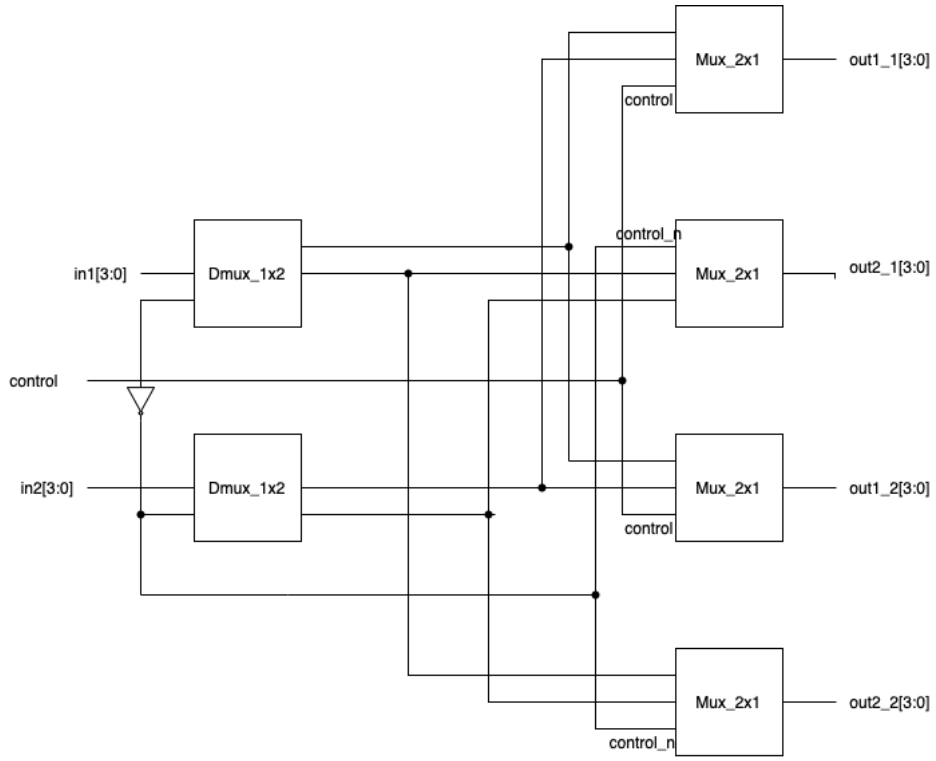


Report

我們用了兩個And和一個Or來組成Xor，再利用lab_1 basic用到的D Flip Flop和Xor還有And來組成Toggle Flip Flop，而D Flip Flop則是由一個inverter和兩個D Latch組成。
因為Toggle Flip Flop是個sequential circuit，所以我們對它的所有狀態都測試不同的輸入(詳見Lab1_Team25_Toggle_Flip_Flop_t.v此檔案中之註解)。
另外，為了測試此Toggle Flip Flop是否只在clk posedge時被trigger，我複製了一遍剛才的測試，並將所有posedge跟negedge時被觸發的事件互換。
最後塞了一段rst_n跟t都是1的狀態，皆無預期外的結果出現。
對於sequential circuit，我沒有一個標準的方法產生出能測試所有可能的testbench。此testbench能測試所有可能狀態的前提是此module是正確的，如若不然，此testbench只能保證在第一個錯誤出現前的探索都是如我們所預期。

CrossBar_2x2_fpga:

Logic Diagram:



(Crossbar_2x2 for fpga)

Report

要燒到fpga板裡面的crossbar_2x2的code和原本crossbar2x2不一樣的地方在output。因為fpga板上的燈有16個，但是原本的code裡面只有8個port接出去，所以我們複製個一個和out1一樣，另一個和out2一樣的出口，來讓我們選擇I/O port。所以圖中的out1_1還有out1_2是一樣的，out2_1和out2_2則是一樣的。

What we have learned from this lab

- ssh and git command
- meaning of XDC file
- testbench writing technique
- debug with the wave diagram
- using nWave

在這次的lab中，我找回了寫verilog的感覺，也沒有了一開始要面對邏設實驗的緊張感。
我們這次的lab除了燒fpga板，都是在CAD server上面完成的，我也學會了如何使用nWave來看波形圖debug。我們也透過修改testbench來測試我們的code，這也讓我們對testbench更加瞭解。因為是小組分工，所以我們決定用github來存所有的資料，不但不會丟失，也可以更容易的整合兩個人所做的作業。

Problems we encountered

- Synthesis failed without any error and warning on vivado
  There’s no many articles for this problem on the internet, but I finally found that the problem is that the username of my computer is including characters without english. And I solve the problem when I change it to english.
- Since some of the group members are not familiar with github, we’ve been facing some issues about push and pull requests, and almost accidently delete all the files in the repository. But after all, we solved the problem successfully and became git masters.

- We’ve met a problem that whlie finishing the code of Crossbar2x2 and Crossbar4x4, we thought that the structure of Crossbar2x2 is wrong, so we change the code of Crossbar2x2, but forgot to change the code for Crossbar4x4. This cost us lots of time to debug, finally we find the bug using bash script. We find out that it’s hard to debug for a verilog code in this problem.

Contributions of each team member

林書辰

- Responsible for the coding of modules.
- Responsible for 20 percent of the report.
- Installation of Vivado.

楊景遇

- Debug and testing.
- Responsible for 80 percent of the report.
- Bash script on cad server.