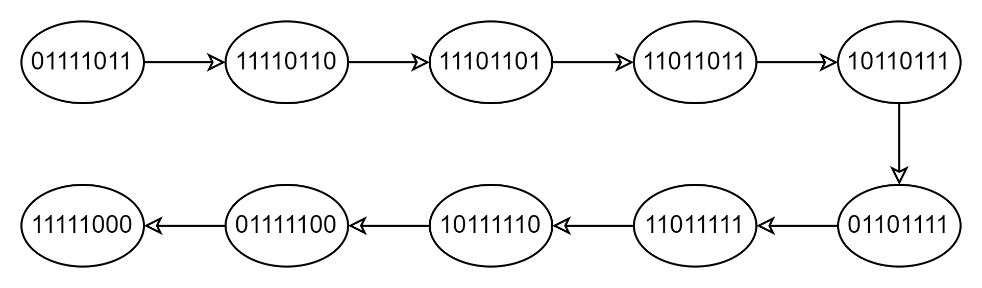
Lab 4 report

組員：110062221李品萱

110062213唐翊雯

1. **Many-to-one linear-feedback shift register (LFSR)**

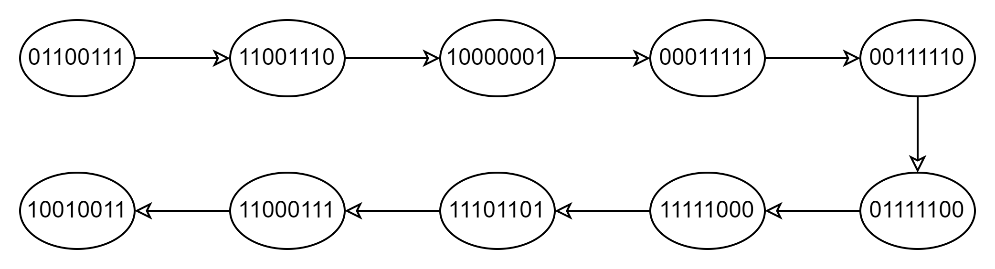
在basic question 3中，我們需要用DFF及xor gate做出一個LFSR，這一題的state transition diagram如下圖。

****

在spec中，我們reset時將DFF reset為8’b1011101，此時若在reset時改將DFF reset為8’b0，則之後的output都會是0，因為DFF單純trigger clock接收及給出對應的output，而xor gate在收到的兩個input都是0的情況下也只會給出0，因此這個情況下電路出來的值不會有任何改變。

1. **One-to-many linear-feedback shift register (LFSR)**

basic question 4 與前一題類似，其 state diagram 如下圖。



同樣的，若我們reset時改將DFF reset 為8'b0，之後的ouput也都會是0，因為DFF會接收0再送出0，而xor gate收到兩個0時output 0，因此最後的結果都會是0。

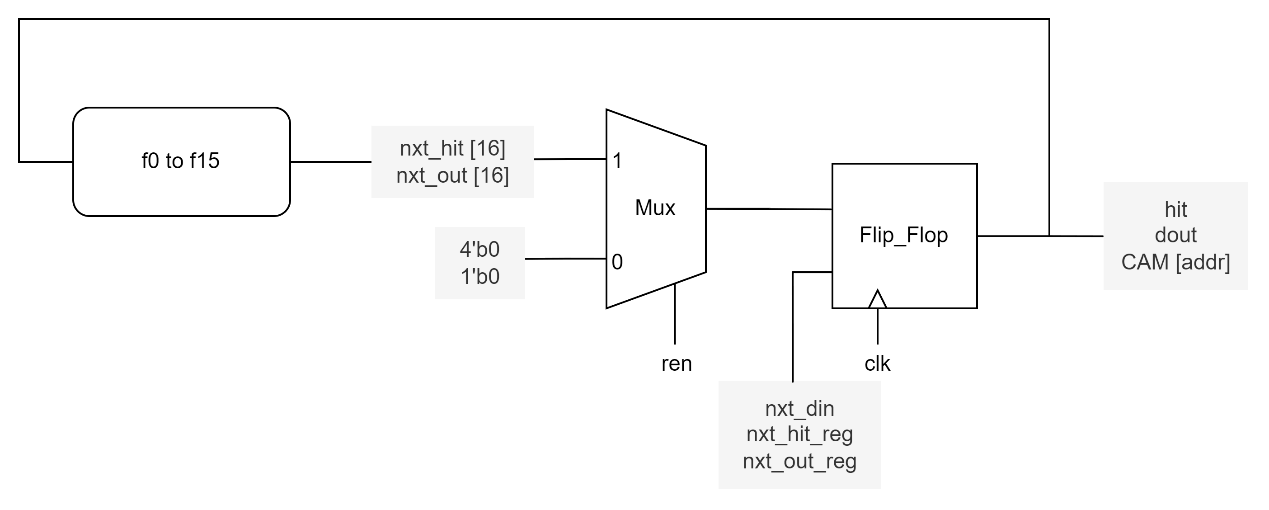
1. **Content-addressable memory (CAM) design**

這題我們需要做出一個CAM，能夠將din存在對應的addr，及output din 對應的addr。

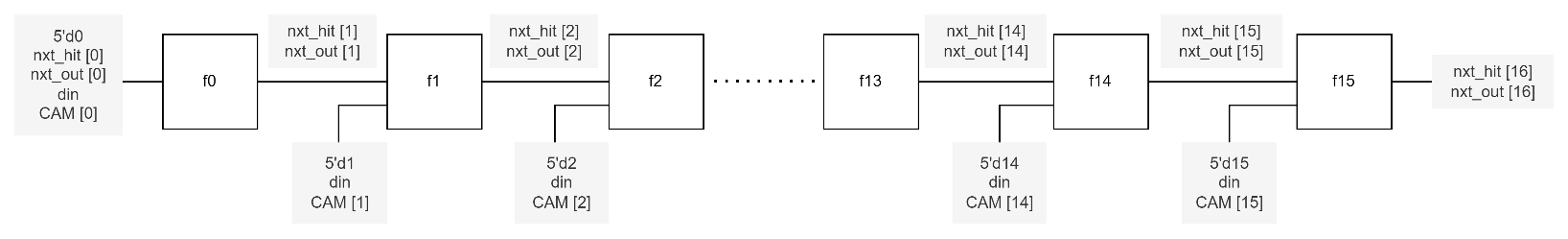
首先，在讀取的部分，我們先看nxt\_din會是什麼，如下圖。



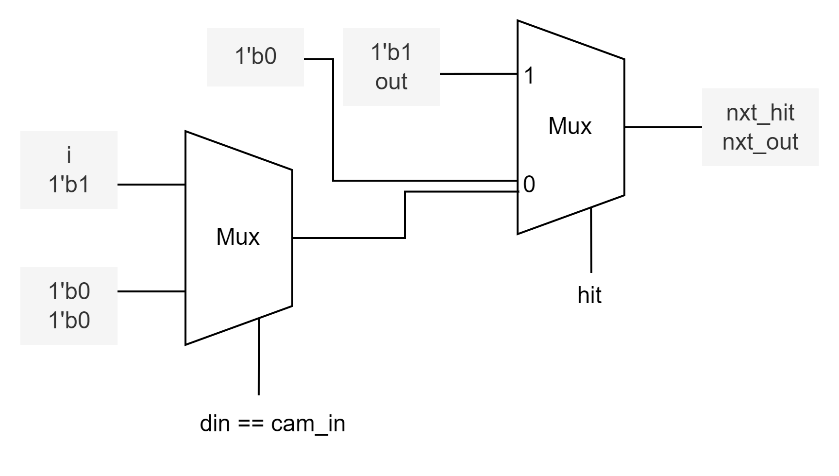
這麼做的原因是，在sequential block中，我們只在wen為1且ren為0時做讀取，此時為了不造成inferred latch的問題出現，我們先看現在是否符合讀取的條件，若符合，nxt\_din即會接din，不符合時nxt\_din則會接該addr位置當前的值。這樣我們在sequential的部分就只要處理ren為1或0對應到的hit與dout。這題電路圖如下。



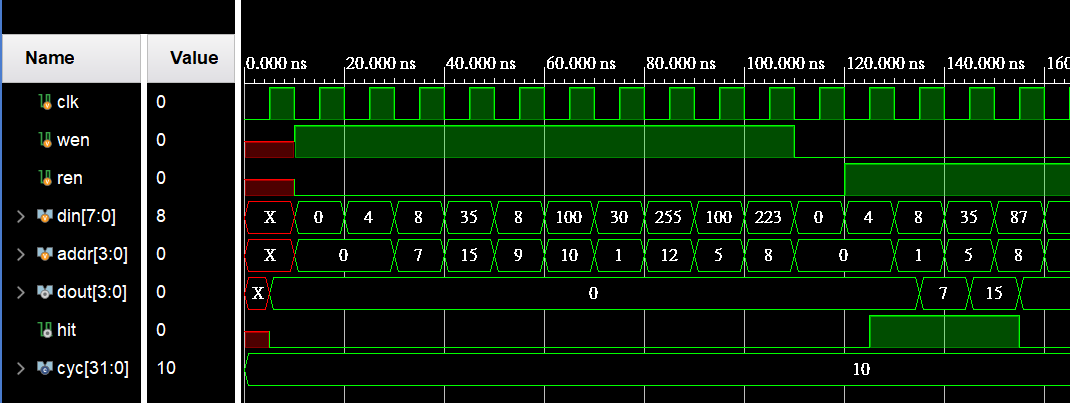
上圖的f0 to f15如下圖。在comparator array及priority encoder的部分原先我們想到的是用for loop讀過整個CAM，但上課時有提過for loop是無法synthesized的，因此我們將for loop的概念實做出來，我們寫了一個for\_loop的module，將16個for\_loop的module接起來，每次將上一個的output作為下一個的input，並送進要檢查的addr及CAM的值，如下圖，我們最後得到nxt\_hit[16]及nxt\_out[16]即為所求。



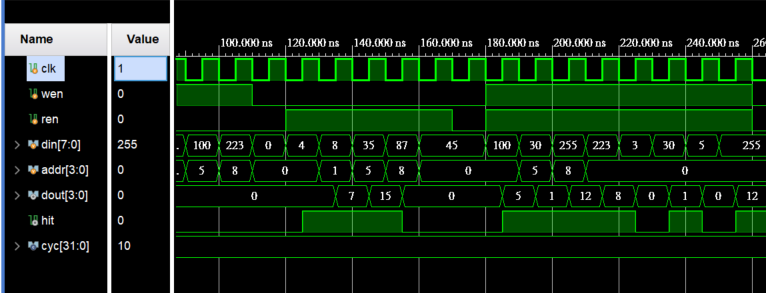
每個for\_loop的設計如下。我們先看現在是否已經找到要讀的值，如果沒有再看現在的值是否是我們要找的。這邊的設計上我們也顧及若有多個match 的addr要output最小的，因此一旦hit值為1，後面便不會再抓對應的addr，整個過程運用了comparator array及priority encoder的概念。



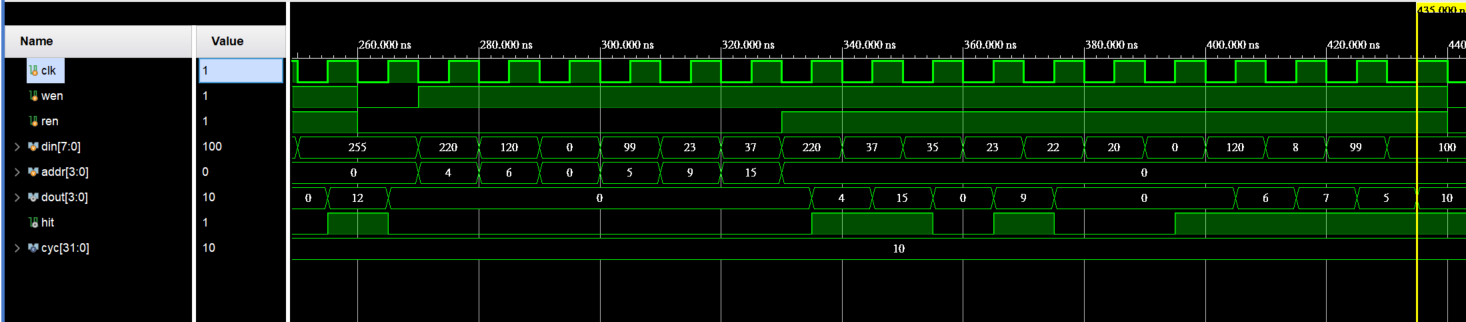
在testbench的部分，我們先做出跟spec相同的圖，初步確認正確性。



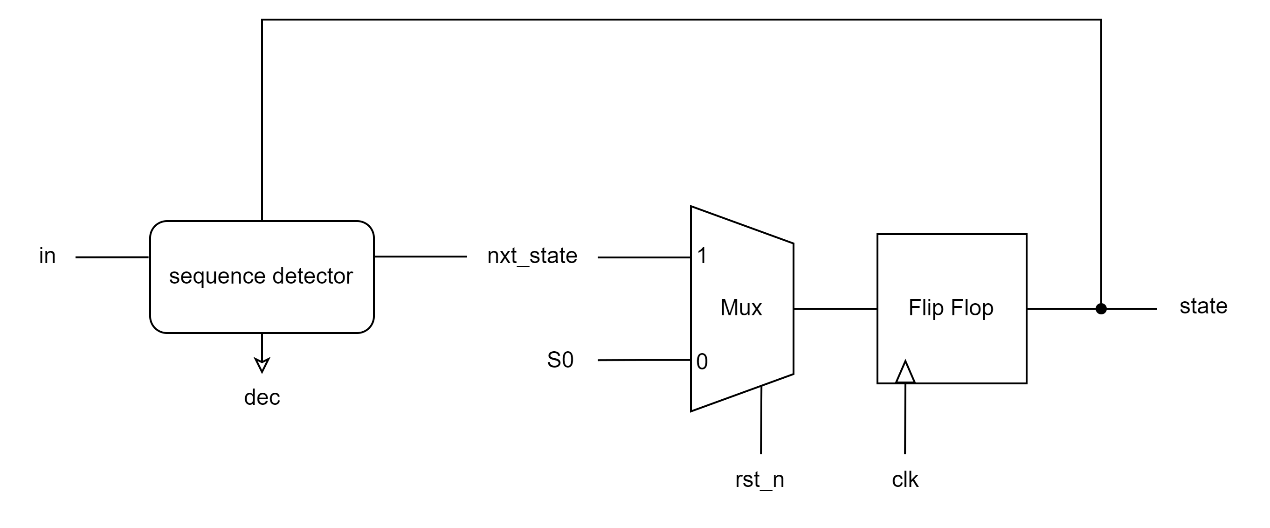
而後我們確認ren及wen同時為1時CAM只會做寫的動作，並且output的是最小的addr，如下圖的din為100會output 5而不是10。



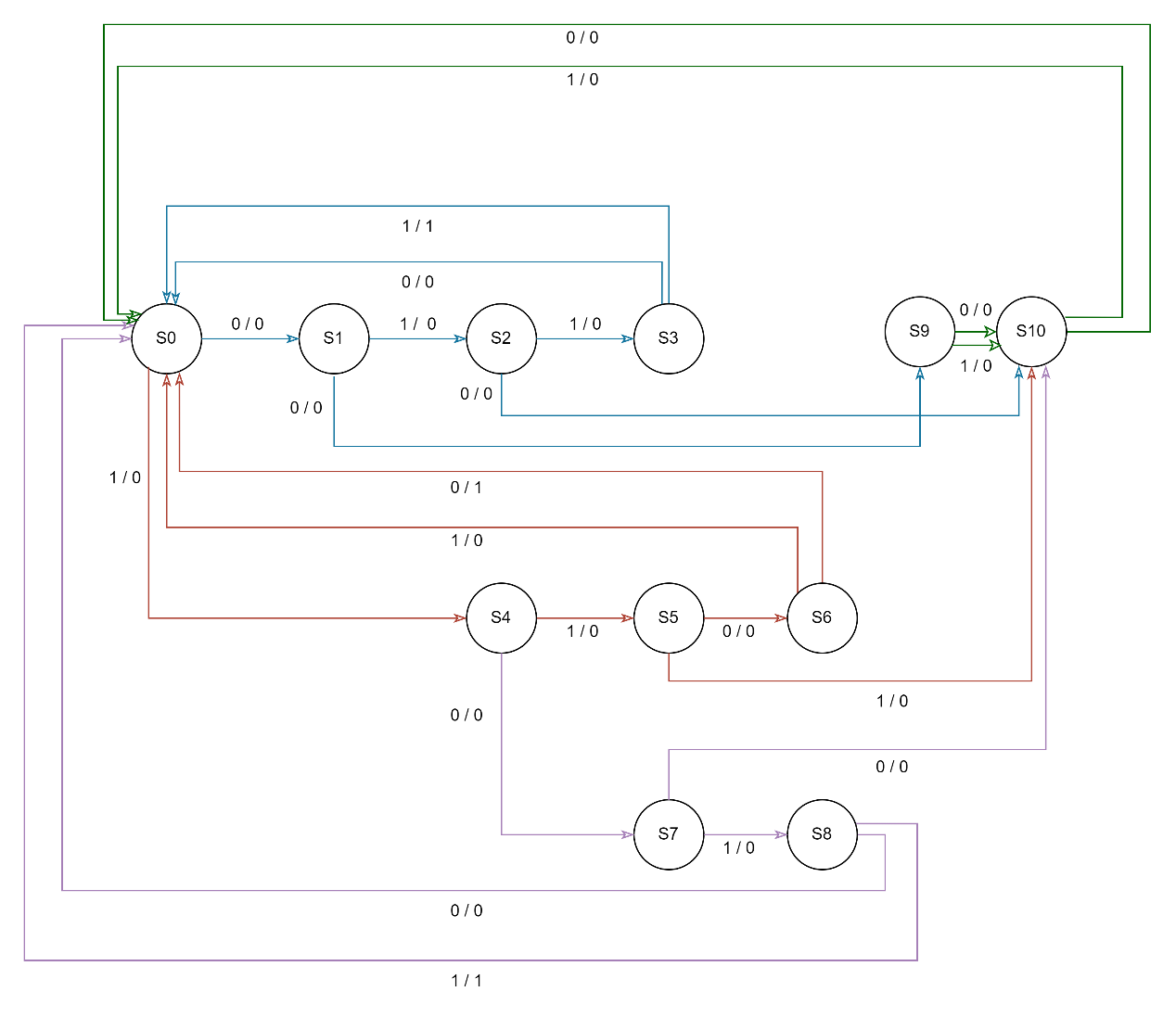
後面我們又再測了一些讀寫的動作，確認CAM及addr的大小有符合題目的要求，且若有新的值存在有值的addr位置，新的值會直接覆蓋掉舊的值，如下圖原本35寫在CAM[15]的位置，但因為這個位置後來寫進了37，因此會dout及hit都是0，而黃線位置的100原本在addr為5及10都能找到，但因為5寫進了99因此din為100的狀況會output 10。



1. **Scan chain design**
2. **Built-in self test**
3. **Mealy machine sequence detector**

****

如上圖（sequence detector代表下圖的state transition），這題我們需要做一個每4個bits detect一次的mealy machine，首先我們先畫出它的state diagram，如下圖。



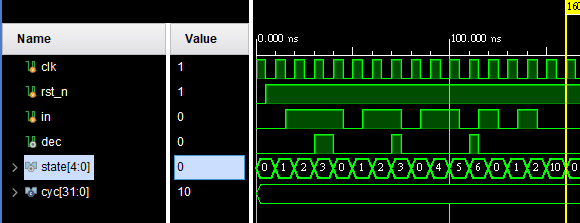
dec為1的條件為，吃到0111，1011或1100三種input sequence，對應的state大致可分成三組：

* 0111 -> (S0, S1, S2, S3)
* 1011 -> (S0, S4, S7, S8)
* 1100 -> (S0, S4, S5, S6)

由於它是4 bits detect一次，因此即使吃到不合法的輸入也要吃完4個bits，所以我們用S9及S10來處理。觀察發現，S0吃到1或0都有可能產生合法輸入，此時如果下一個輸入是不合法的，我們會需要再經過兩個state再回到S0，而S9就是在做這件事，S10同理，當我們吃到第三個輸入發現不合法時我們會需要一個state去吃最後的input，再回到S0。

在設計state transition的部份我們先寫出三組合法sequence各自的state變化，發現他們都可以共用S0，而1100及1011可以共用S4，因此處理合法sequence的部份我們只需要S0到S8即可，接著依照input對應的state接起來即為所要的state diagram。

Testbench的部分我們先確認能得到與spec上相同的結果，為了方便debug，我們將state output出來，如下圖。



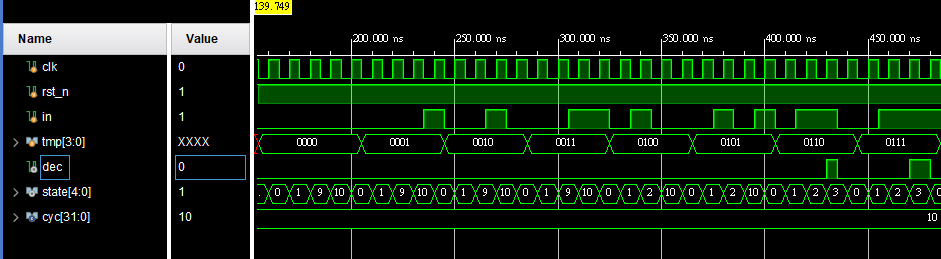
在第一次跑到S3時，可以看到dec為1的狀態持續了不只一個clock，這是因為它是一個mealy machine，前面起來的1是state change的1，但此時clock還沒起來，又收到1，而input變output就會變，所以又會output 1。

而後我們枚舉所有可能的input sequence，code如下圖。

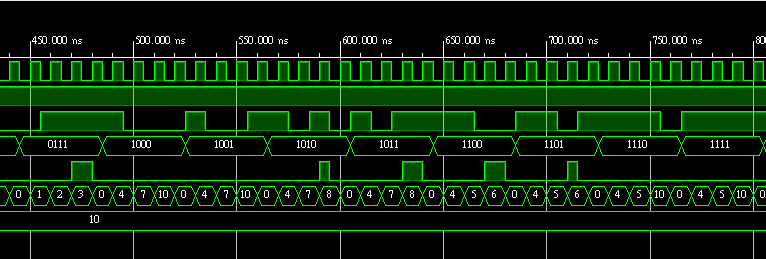
一張含有 文字, 電子用品, 螢幕 的圖片

自動產生的描述

waveform如下面兩張圖，其中，在tmp為0110出現dec為1是因為它在s3變成下一個state之前收完0111。



下圖中1010及1101的區間看到的dec為1原因同上，使他們為1的sequence分別為1011及1100。



1. **Summary**

這次的lab我們學到了finite state machine的實作及應用，也持續熟悉memory的操作。

這幾次lab的操作下來，我們在寫硬體的時候更能夠以硬體的方式思考及設計，例如這次的CAM我們一開始想到要跑過整個memory找答案，但它不像軟體可以直接跑迴圈，因此我們去思考我們的電路圖應該長怎樣進而做出對應的設計，也會去想麼避免inferred latch等等會使電路沒辦法正確接好的問題。

在Mealy machine sequence detector的部分，一開始寫的時候以為它是每次多往後看一個bit所以多花了一點時間，後來發現與spec的圖對不上才又改正，這題讓我們學到了從設計state transition到設計出對應的mealy machine。

1. **Contributions**

* **Code:**

Content-addressable memory (CAM) design by 唐翊雯

Scan chain design by 李品萱

Built-in self test by 李品萱

Mealy machine sequence detector by 唐翊雯

FPGA - 1A2B game by 李品萱

* **Report:** 各自描述負責的題目