Logic Lab

Final Project

Report

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
| 系級 | 114 | 114 |  |
| 學號 | E24106084 | E24104278 |  |
| 姓名 | 謝宗翰 | 潘品叡 |  |

1. **Screenshot the pass screens of 2 testbenches**

(Please use “**testbench1\_v2**” & “**testbench2\_v2**”)

* Testbench1 v2

一張含有 文字 的圖片

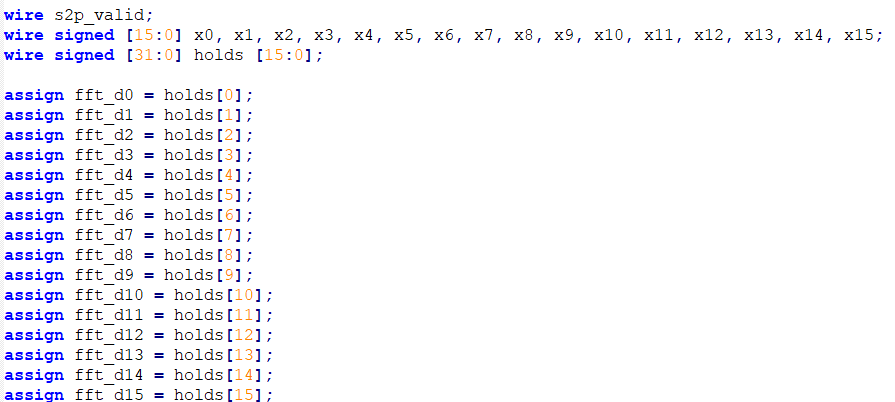
自動產生的描述

* Testbench2 v2
* 一張含有 文字 的圖片

  自動產生的描述

1. **Describe how you design and implement each component**
   * FAS

這部分主要就是把下面四個module包起來，宣告那些也都按照該module即可，在module之間會宣告一些wire來傳送資料。



* + FIR

宣告一個reg的count來計算data進來的次數，並宣告31個16bit的DFF來儲存資料，並用sequential電路去做出data shift的功能。

一張含有 文字 的圖片

自動產生的描述

將每個DFF的data乘上助教提供的係數，來達成濾波的功能，因為不確定電路上有沒有程式中+=的功能，所以開32個add，一層層堆上去。而add之所以是35bit是因為他是{1 sign, 7 int, 8 float}\*{1 sign, 3 int, 16 float} = {1 sign, 10 int, 24 float} 總35位數

一張含有 桌 的圖片

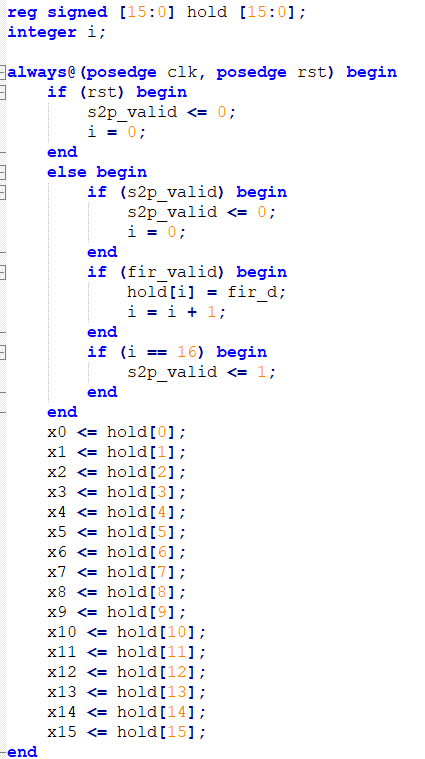
自動產生的描述

由於最後輸出的fir\_data要是{1 sign, 7 int, 8 float}的形式，所以要做bit縮減，而根據add的形式，我們可以得知要整數要從第30位開始取，最後還要檢查signed bit是否為負，如果是那就加一(測出來如果不加答案會錯，猜測可能助教給的測資是一補數)



* + S2P

利用sequential電路製作一個類似for計時器的功能，將每個時間的data個別存放在hold裡面，然後給到output上，並在i==16時把s2p\_valid拉成high，下個clk如果檢查到s2p\_valid為high就先把他拉low，以此保證他只回high一個clk。



* + FFT

這個module的功能比較繁雜，其實要就是在處理數值運算的bit跟signed問題，我挑幾個地方來說明，第一個是要記得在儲存input進來的數時，在後面補上16bit的0做為起始的虛數。

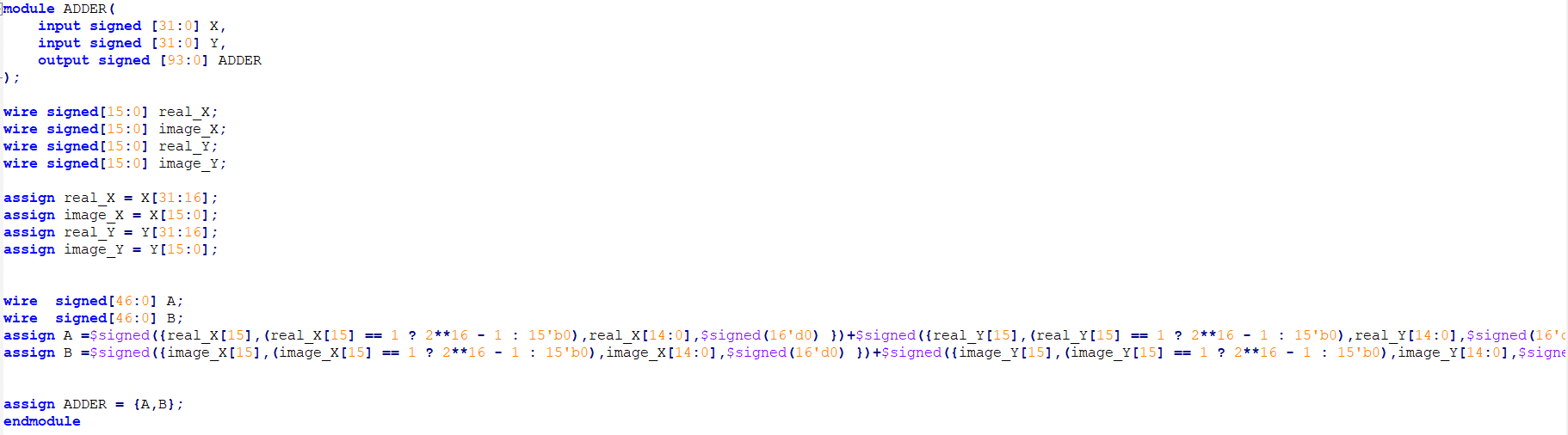
一張含有 桌 的圖片

自動產生的描述

再來的接線應該是最複雜的，為了簡化電路，創立兩個module來處理數值的運算，分別為ADDER(負責把兩個複數相加)跟COUNT(負責把兩個複數相減並且與助教給的係數相乘)，以下分別說明這兩個要注意的點

ADDER

將進來的兩個複數，先分別拆成實部跟虛部，並宣告A來儲存相加後的實部B來儲存相加後的虛部，但因為要顧及COUNT運算後造成的bit數增加，這裡的output也要變得跟COUNT的bit數一樣，所以integer的部分補15個0或1(看signed bit)，而float則直接補16個0。Assign A,B的時候，外面兩個$signed()不可省略，不然他不會判斷為有號數(因為()裡面本身就不為有號數了，實作的時候被這個搞到過= =)



(此圖有些裁減，詳細請看原始碼)

COUNT

將進來的兩個複數，先分別拆成實部跟虛部，一樣宣告A來儲存處理完後的實部B來儲存處理完後的虛部，不過這次要多宣告一個sum先來把進來的兩個複數相減並儲存，讓後面比要好運算。

一張含有 文字 的圖片

自動產生的描述

總共分成四層，由於每層的bit數都會變多，所以ADDER跟COUNT也是每層用一個，像第二層就會變ADDER\_1跟COUNT\_1這樣，功能都一樣，只有bit數的多寡差別，每層的bit計算如下

Layer1:

{1 sign, 7 int, 8 float} \* {1 sign, 15 int, 16 float} = {1 sign, 22 int, 24 float} 總47位數

Layer2:

{1 sign, 22 int, 24 float}\*{1 sign, 15 int, 16 float} = {1 sign, 37 int, 40 float} 總78位數

Layer3:

{1 sign, 37 int, 40 float}\*{1 sign, 15 int, 16 float} = {1 sign, 52 int, 56 float} 總109位數

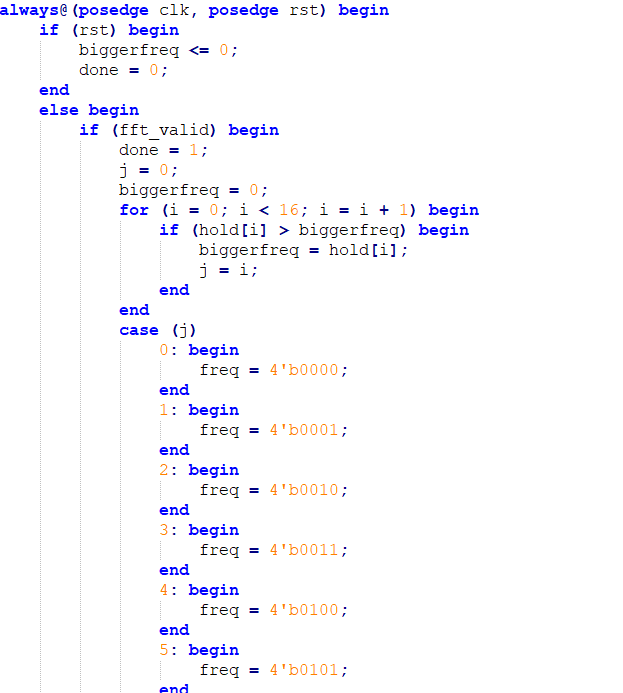
Layer4:

{1 sign, 52 int, 56 float}\*{1 sign, 15 int, 16 float} = {1 sign, 67 int, 72 float} 總140位數

在Layer4的時候，由於是最後一層，所以輸出的bit數要再拉回{{1,7,8},{1,7,8}}的形式，所以會在該層的ADDER&COUNT就做處理，如下



* + Analysis

進到這邊的數值儲存前面都有用提過了，主要說明如何挑選主頻率的方法。其實說起來跟程式的觀點比較像，若在For迴圈中當前頻率大小比儲存的還大，那就把i值記錄下來，最後利用switch把i換成binary的形式。

1. **Experience (實驗心得)**

**謝宗翰:**

**Final-Project果然是跟平常課堂教的module是不同層級的，當初剛看完影片時超級一頭霧水，大概知道當中每個module的功能，不過對於其中的I/O的物理意義跟為何可以這樣實作(濾波之類的)不是很了解，還好助教都蠻快就回信並且講得很詳細(雖然有些還是沒有了解，畢竟好像之後的課程才會教到原理)，在跟助教和同學激烈溝通完後才開始知道大概要怎麼寫，不過寫完後發現數值處理的部分錯的很離譜，因為沒有考慮到signed的宣告與運算時造成bit數量的增減，之後debug的時後才發現有許多insecure的地方，而這次在debug的時候不太能從波行圖上看出哪邊可能是出了什麼錯，整體的感覺跟verilog比較沒有關係。對於這次的Final-Project，我認為比起coding，了解如何降噪的過程反而讓我更為印象深刻。**

**這學期的邏輯實驗課程也到了尾聲，從一開始對於這個領域完全不了解，現在瞭解了Latch,flipflop這些基礎module的線路及功能，還有verilog裡面wire&reg,sequential&combinational等等的區別，配合邏輯系統課程在moore&mealy machine和state-table上的說明，我認為這學期的邏輯課程確實讓我在這方面有了很大的成長，可能沒到完全理解，但我會給自己打大概80分吧。回憶這學期的課程，雖然好像每次上實驗課時都常常處於資訊量爆炸跟verilog打不完的情況，但因為助教們活潑的教學方式，並且在我們遇到問題時都會用心幫我們解決，讓我始終感受到課程是有趣且新奇的。謝謝助教們這學期的教導及幫助!!**

潘品叡:

三周前所公告的final project是本堂課的最後一份作業。在一開始時，我與組員抱著許多問題都不確定的狀態去設想寫code，然而，在經過幾次寄信詢問助教以及與各個組別的人討論最終所有問題都得到了答案。這次的題目難度之大使得我們花費了許多時間處理，包括debug。不過，寫code何嘗不是如此？因此，儘管難度很大，但仍很高興能在最後一次的作業中學到不少東西，同時也為這一學期畫上了句號。期許自己將來可以更精通verilog，完成更多更複雜的任務。