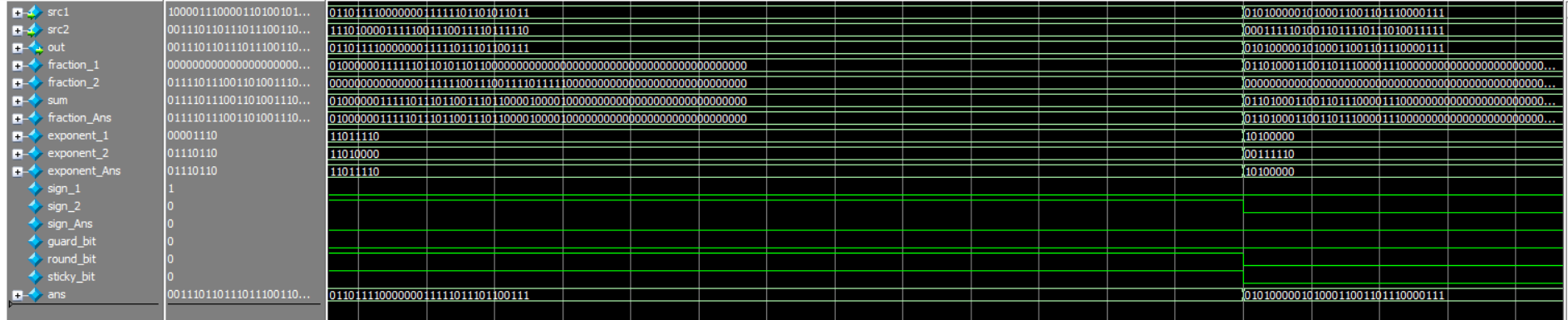
**Computer Organization 2019**

**HOMEWORK 3**

系級: 資訊110 學號: F74065042 姓名: 李浩榮

**實驗結果圖:**





**程式運作流程:**

基本上都是數值,就不會有太多變化,除了sign\_bit,若正+正為正,正+負(正>負)為正,正+負(負>正)為負,負+負為負的變化,以及guard\_bit,round\_bit,sticky\_bit的變化看得出來,其他的就都是數值

=================================  
流程如下:

1. Loading

先載入src1和src2,然後分裝成fraction\_1,fraction\_2,exponent\_1 ….

1. Preprocessing

這裡比較特別的是為了處理exponent為0,且fraction不為0的時候的值,所以直接把exponent變成1

再來就是根據exponent和fraction的case去設定fraction\_MSB左邊的bit為0或者1(其實就是為了0.XXXX 或者1.XXXX的情況)

1. Special\_case

當其中一個輸入為0的時候,回傳另外一個值

1. Align

判斷哪一個exponent較大,shift較小的fraction (exponent\_Bigger-exponent\_Smaller)個位,然後得出exponent\_Ans的值

1. Add\_Significands

判斷正負號

1. Overflow

當超過fraction的23個bit外的第24個bit到第25個bit的時候就是算Overflow.

01.XXXX (Overflow = X)

1X.XXXX(Overflow = O)

就是判斷第65個bit是否為1否則shift right,exponent 加1

1. Normalization

Normalization的意思是第65個bit要為1(也就是想辦法變成1.XXX的情況)

如果是(1X.XXXX)的情況,就shift right 1位(exponent要加)

如果是(01.XXXX)的情況,就不用shift

如果是(00.XXXX)的情況,就要判斷fraction中的23個bit中是否有1,有的話就一直shift left到第65個bit(exponent要減)

1. Convert

整合成Output,以及處理一些特別情況

=============================================

**心得**

心得就是寫了很久但還是很開心~

先說遇到的問題和解決的方法好了:

* 不能用state來劃分區塊,因為這不是吃clk的,所以只能擠在一起.如果能夠區分開來的話,對於程式的整理以及debug也比較容易,這是一個verilog的好習慣
* 學到了floatingPoint的overflow到底是什麼情況,才算是
* bugs的變化是從1500~1100~110~11~0 ,由此可見其實程式並不難,只是概念要對就行了
* 其中我之前概念以及流程有點不一樣導致bug會出現,但是調換了就解決了.例如我之前是add\_Significands後做round然後再做normalization,但是會造成就是normalization前的round和之後的round加的第42個bit的相對位置對於fraction\_Ans來說是不一樣的
* 還有就是當exponent為0,fraction為non-zero的時候,就直接把exponent變成1的這種測資比較特別,如果沒聽清楚或者理解恐怕一輩子都無法理解xD(其實PDF的那個表有寫了-126了,所以其實exponent是1的意思)
* 還有就是要記得加1或者0來表示1.XXX或者0.XXX的情況. 根據exponent為0的話,就是0.XXX,non-zero的話(1~255)就是1.XXX .後面再去用來做overflow和normalization的.這很重要.
* 我這裡找sticky\_bit是用while迴圈去找fraction後的40個bits中有沒有1,但是建議不要這樣做,因為對於硬體來說會很耗時,for迴圈也會有一樣的結果.因為硬體對於for和while的理解是同時展開並且同時進行計算,所以gate的使用量會很大,就不太好的做法…

基本上就是這樣,其實不難,只要過程和shift要理解清楚就可以了.個人認為寫這種verilog還蠻有趣的~很期待下一次作業 :)