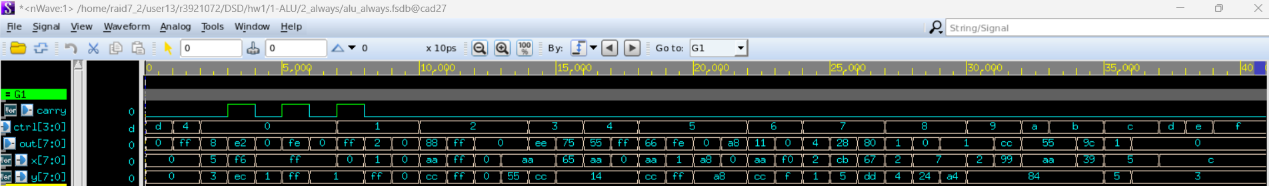
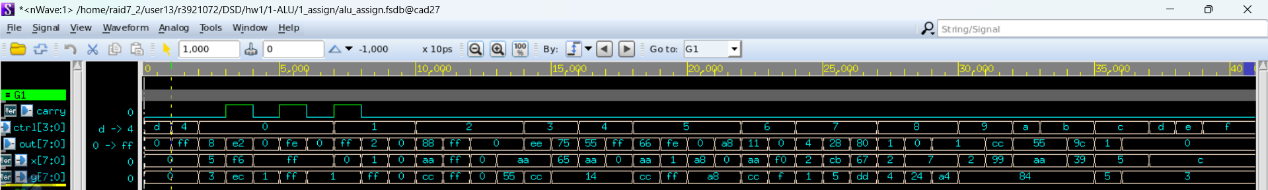
**DSD\_Report\_HW1**

R13921072 何家祥

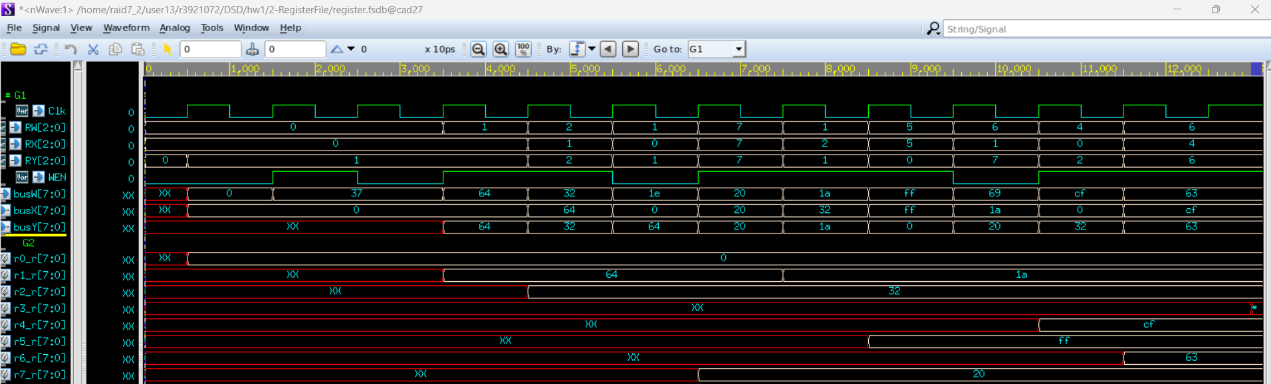
1. **ALU (assign & always)**



由於這兩份所設計的東西是一樣的，只差在使用的語法不太一樣，因此放在一起說明，Testbench的設計根據ctrl，一個一個檢查，先檢查簡單的加減法(負+正、負-負等等)有沒有運行，並丟一些會overflow的數值來檢查carry的值是否正確。

在基本邏輯閘的部分，將xy帶入全1、全0、完全相同、完全相異等方式來檢查。至於左右移動以及相等功能，我選擇隨機帶入一些數字進行檢查，並且在這些部分，我都有監控carry的部分是否有錯誤的把訊號拉起來。在最後也檢查了剩餘未指定的控制器，是否在xy為任意數時依舊能保持輸出為0。

1. **8x8 Register File**



至於暫存器設計的部分，由於輸入訊號並沒有rst或是rst\_n，並且題目也沒要求要先把暫存器內容初始化，因此從波型圖能看出在還沒賦值前，每個暫存器都會是unknown。

設計testbench部分則是，首先確認r0是否始終維持在0，不管輸入input為何，都應當保持在零，然後檢查當WEN為0時(只讀模式)，是否會錯誤的寫入資料，然後檢查在上一個週期寫的資料是否有正確寫入，調換RX, RY看是否也是按照所設定的值，嘗試再次寫入同一個暫存器，是否有成功覆蓋成新的值。

1. **What I found**

在寫ALU的時候比較簡單，僅需依據控制器case by case進行描述，不過由於他有4位元的case，因此我將它以最高位元的不同拆分成兩部分，期望它能降低路徑延遲，不用每次判斷ctrl時都要全部位元都做比較。除此之外，這次的加減法器是要當作有號數運算，因此我用擴展最高位的方式達成保留進位的效果。

在寫暫存器的時候，我一開始只用一個reg同時當作readwrite，結果發現讀取的結果可能會發生錯誤，因此才用原本的tips分為兩組，並時刻(combination)將reg\_w寫入reg\_r中來達到暫存器的效果。

並且在debug的過程中，最常遇到的問題是會漏寫條件，在跟同學討論後，決定盡量將同一個變數寫在同一個block裡面，就比較好檢查特定變數的問題。除此之外，我在TB多設計一個boolean1變數’fail’，當輸出訊號不符合預期時，就將其賦為1，因此在檢查錯誤時，只要看波型圖fail訊號拉起來的位置，就知道是錯在哪個運算了。