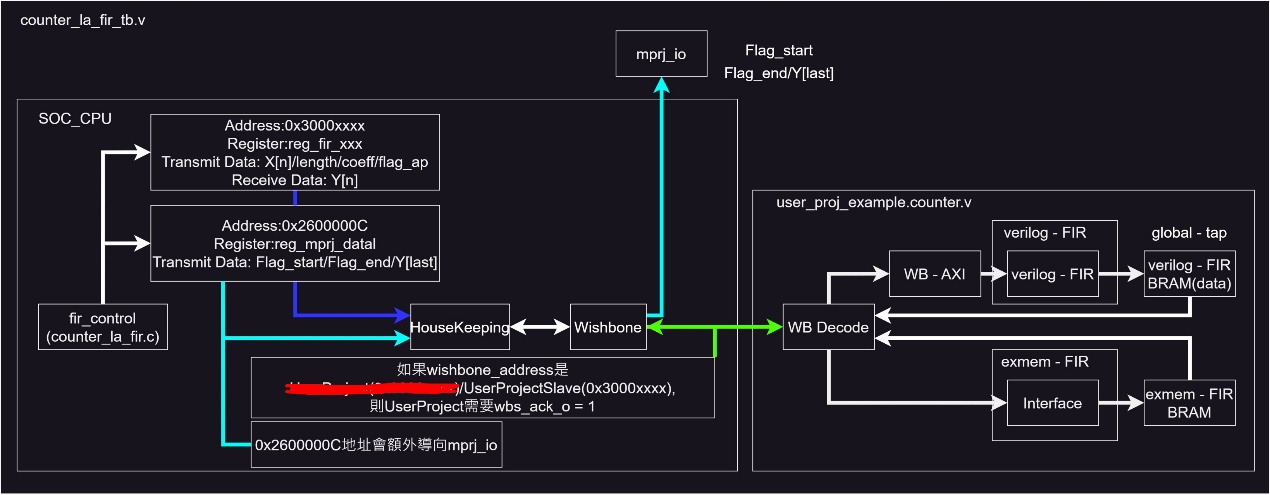
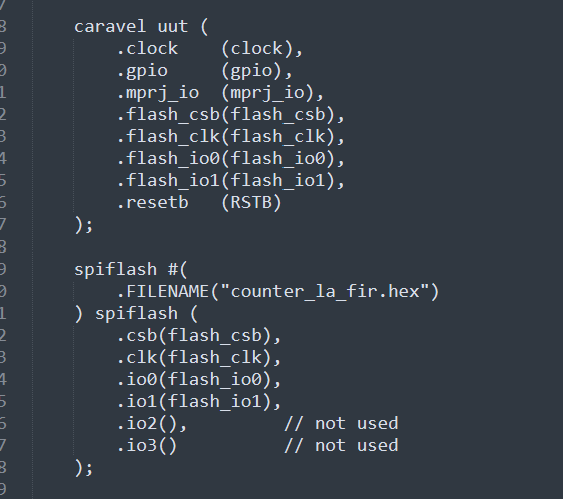
**SOC LAB 4-2**

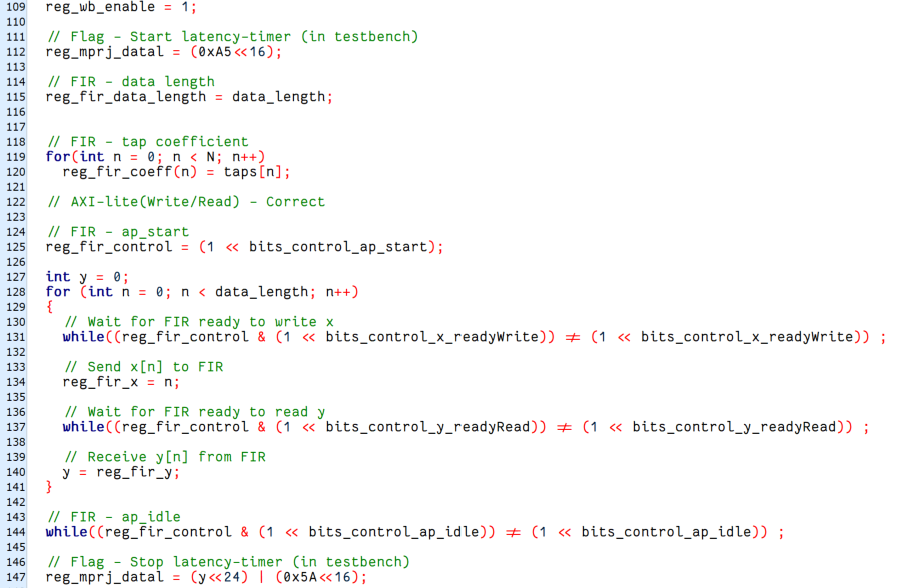
* **Design block diagram**

****

* **The interface protocol between firmware, user project and testbench**
* Testbench: 透過flash control 將資料傳達給CPU執行firmware code



* Firmware(參考fir.c):



程式前面定義各個ip的位置，以下解說程式  
**112 :**

reg\_mprj\_datal 有32bit，前16 -> CPU ; 後16 -> user\_project

因為在cpu運作所以需要左移16bits

**125:**

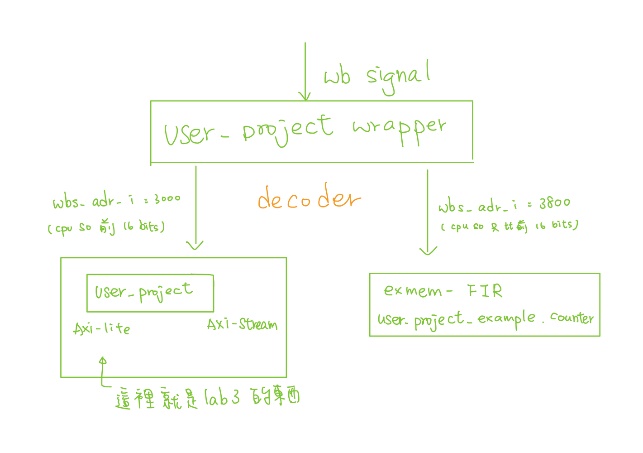
ap\_start signal

**127~147 :**

用bits\_control => ap\_start , ap\_done, ap\_idle

這邊的訊號會透過mprj傳輸到fir.v

* **User\_project**

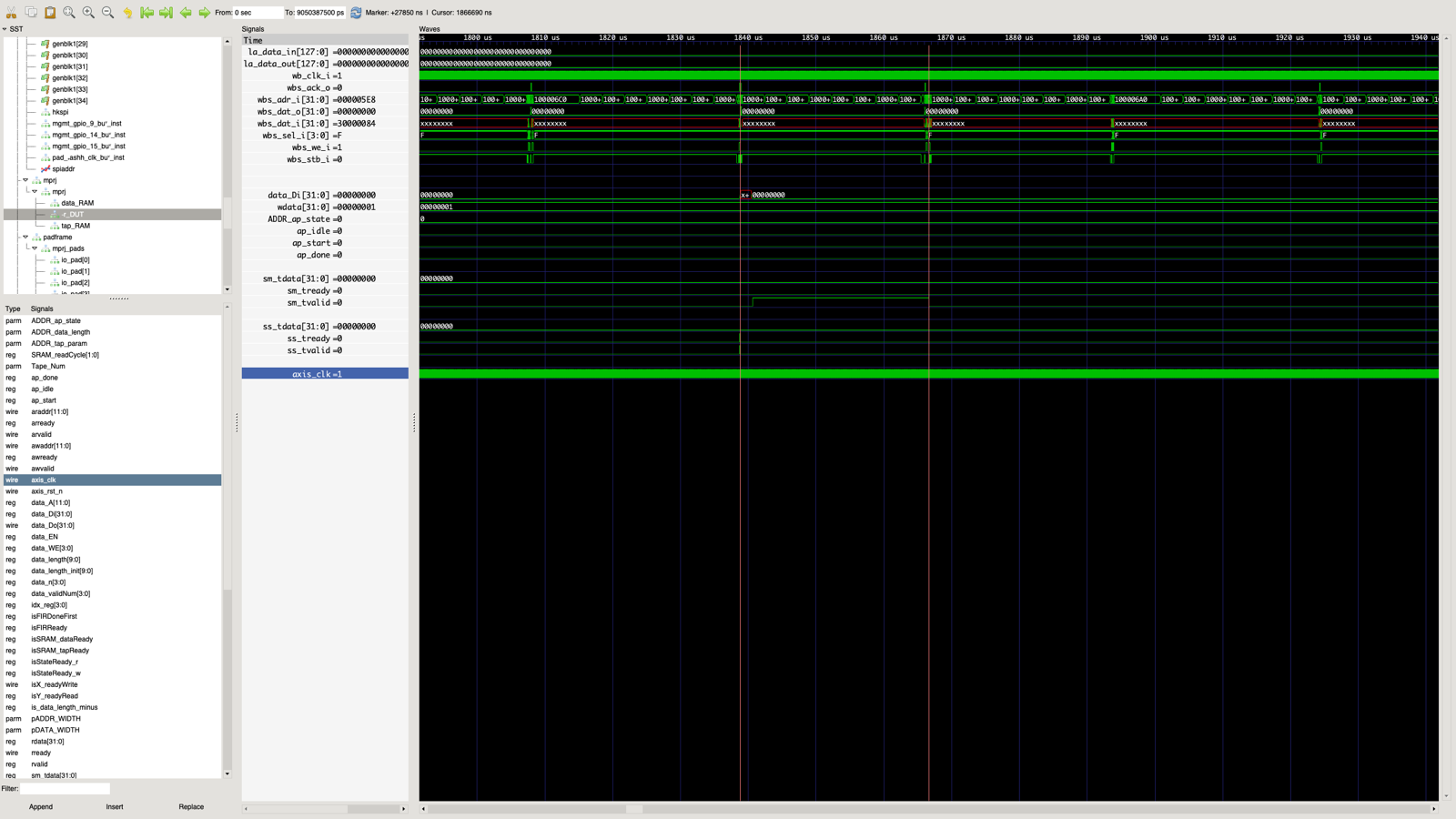


* **Waveform and analysis of the hardware/software behavior.**

**Software :**

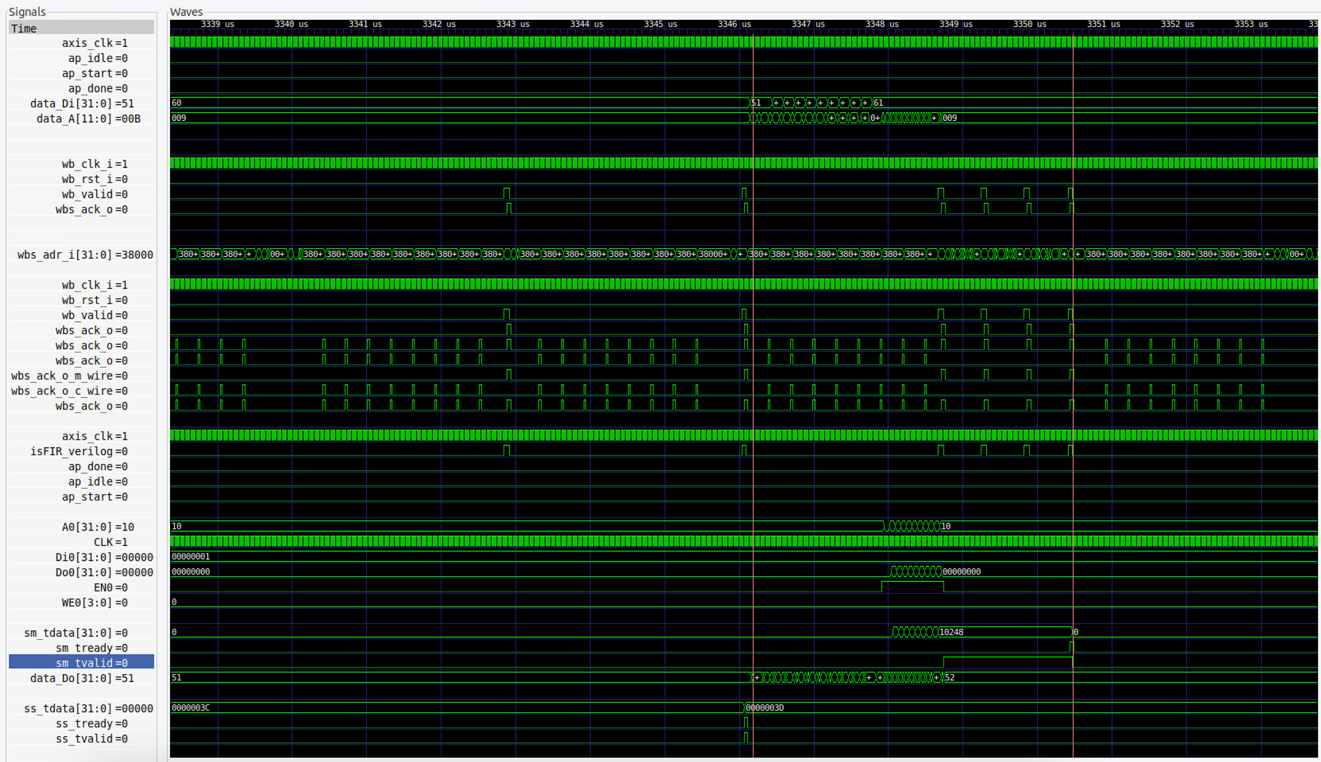
1. 將Firmware 放在SPI Flash 運行

從傳送X[n]至接收Y[n] 約歷經26 µs

****

1. 將Firmware 放在User RAM運行 (0x3800\_xxxx)

從傳送X[n]至接收Y[n] 約歷經4.5 µs

****

1. 結論

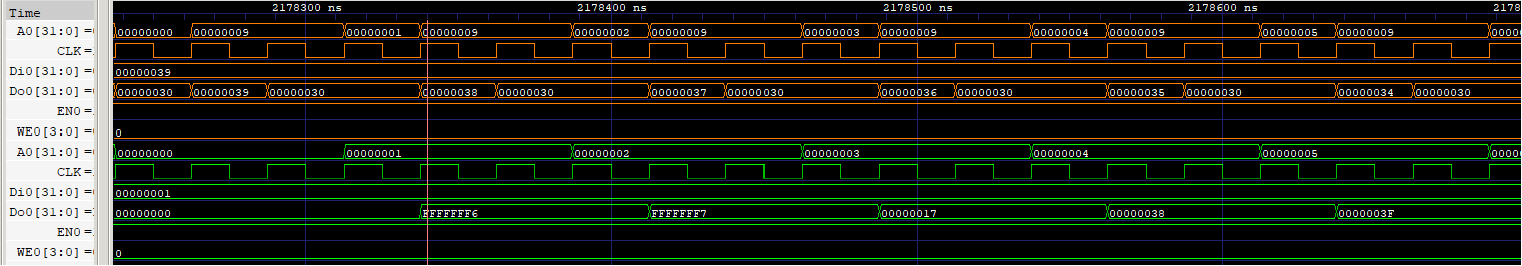
由於Firmware主掌資料流的控制權，意味著Firmware的效能限縮硬體的運算速度，從接收到發送歷經上百個cycle，皆超出計算單筆資料的運算時間，因此我們使用Lab3中資源使用最少的FIR作為本次實驗的硬體端設計。

**Hardware:**

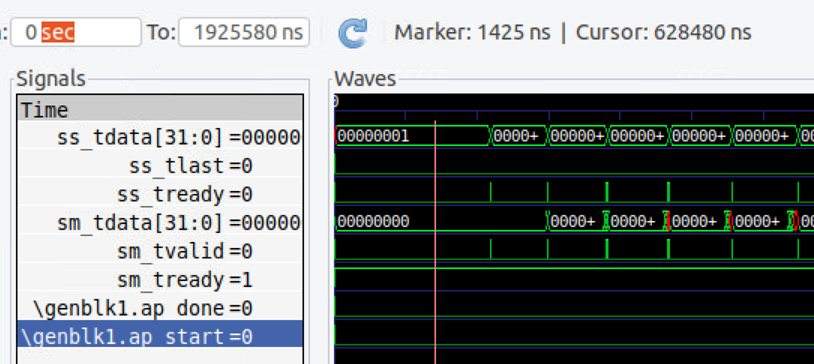
* **0x3800 for exmem-FIR**

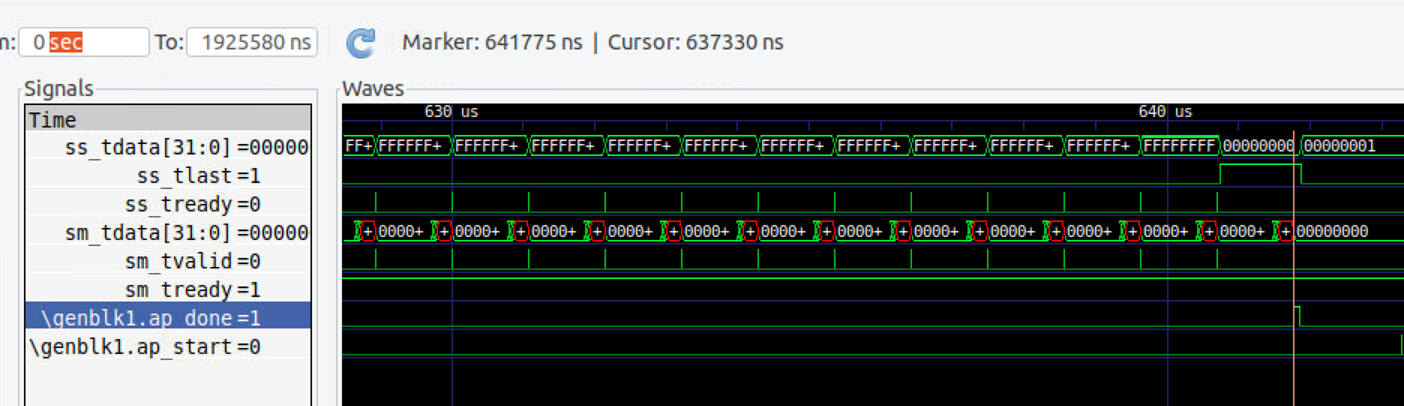
****

* **0x3000 for FIR-AXILITE(BRAM) & FIR-AXISTREAM(BRAM)**

****

* **What is the FIR engine theoretical throughput, i.e. data rate? Actually measured throughput? data rate? Actually measured throughput?**

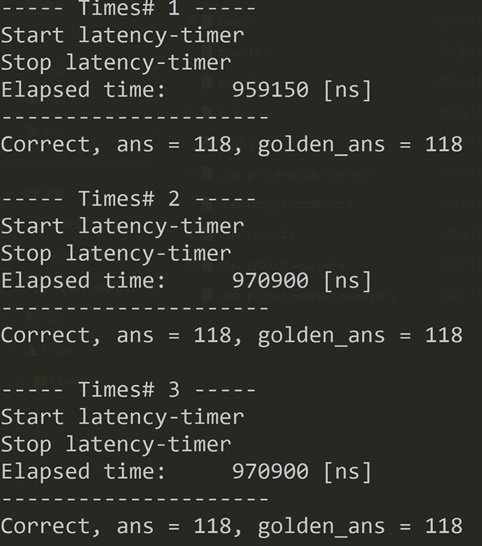
Ap\_start assert at 1425 ns ,and Ap\_done assert at 641775 ns****

****

以Lab3結果而言，最理想的情況下可以在640350 ns 內完成 600筆資料的運算（每筆資料相當於11次乘法，10次加法）。

Data rate = 9370000

Throughput = 196,770,000

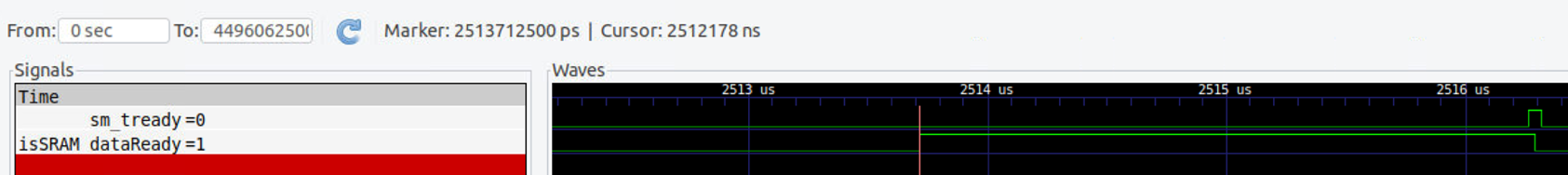
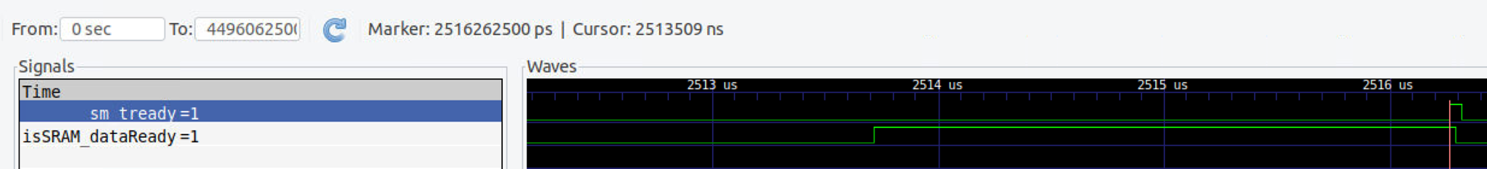


但是實際上量測到的結果，在970900 ns內只能完成64筆資料的運算

Data rate = 61800

Throughput = 1,297,800

* **What is latency for firmware to feed data?**

****

**Latency : 2,550 ns**

* **What techniques used to improve the** **throughput?**

由於目前效能受限於Firmware，可以嘗試以下方法提升throughput

1. 增加資料傳輸效率，以本次實驗為例，單筆x[n]不會用到32bits，所以可以合併多筆x[n]一次傳送，假設x[n]的範圍為1~64，代表只需要使用到7bits，可以讓data[31:25]代表x[n]，data[24:18]代表x[n+1]，以此類推，Y[n]的部分也可以仿效。
2. 增加Input/Output Buffer，如此一來可以配合CPU節奏收發資料。
3. 加快 Caraval SoC運作頻率

* **Does bram12 give better performance, in what way?**

多出更多RAM能讓運算更有彈性，減少control負擔，但在本次實驗中成效有限，因為硬體做再快也會受限於韌體速度。

Bram12多了一個記憶體位置，可以當作Buffer使用，也就是說當FIR還未完成當前運算的期間，CPU就能先將下一次運算需要的x[n]寫入，並且不會覆蓋掉可能需要用到的資料。

* **Can you suggest other method to improve the performance?**

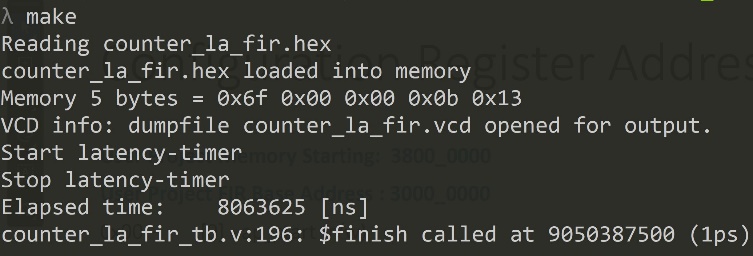
進一步推廣單次傳送多筆資料的想法，並且推廣至更廣泛的數值，譬如使用BF16格式進行資料傳輸，單次傳輸可以傳送2筆資料，而且數值範圍廣泛，雖然增加硬體負擔，需要多進行一個解碼、編碼的任務，但以上百個cycle內完成的前提下硬體依舊游刃有餘。

使用Share memory進行資料傳輸，CPU不必等待FIR接收資料，可以事先將多筆x[n]資料儲存在共用記憶體，等到FIR需要時再讀取，反之FIR也能將運算完成的Y[n]存放在共享記憶體，等CPU有空再去讀取。

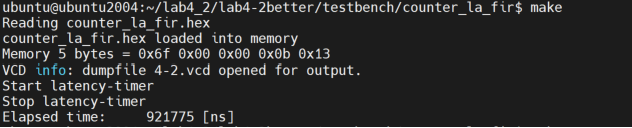
* **Any other insights ?**

對比將Firmware存放在SPI Flash 以及 User RAM之運行時間

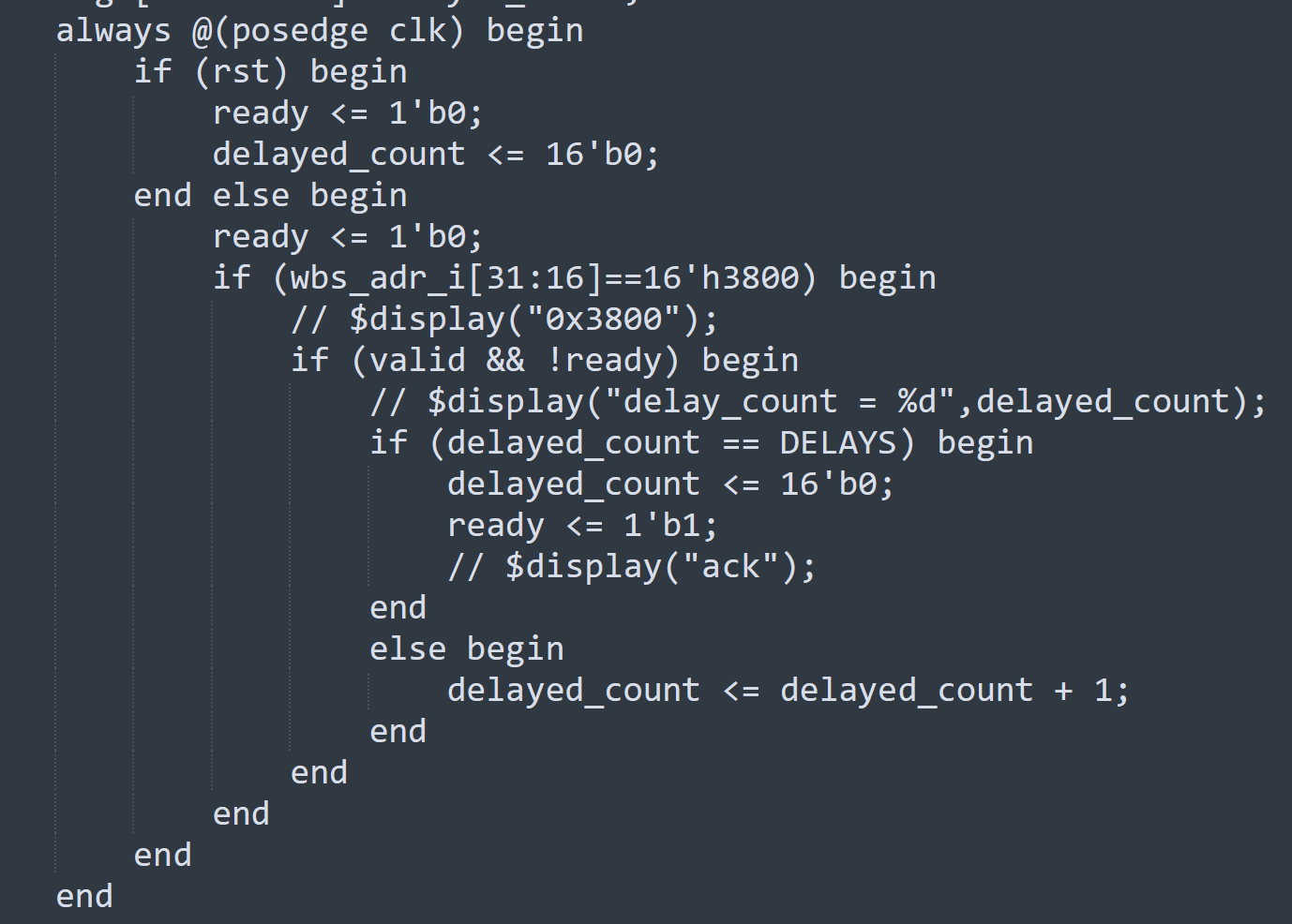
Initial :8063625ns



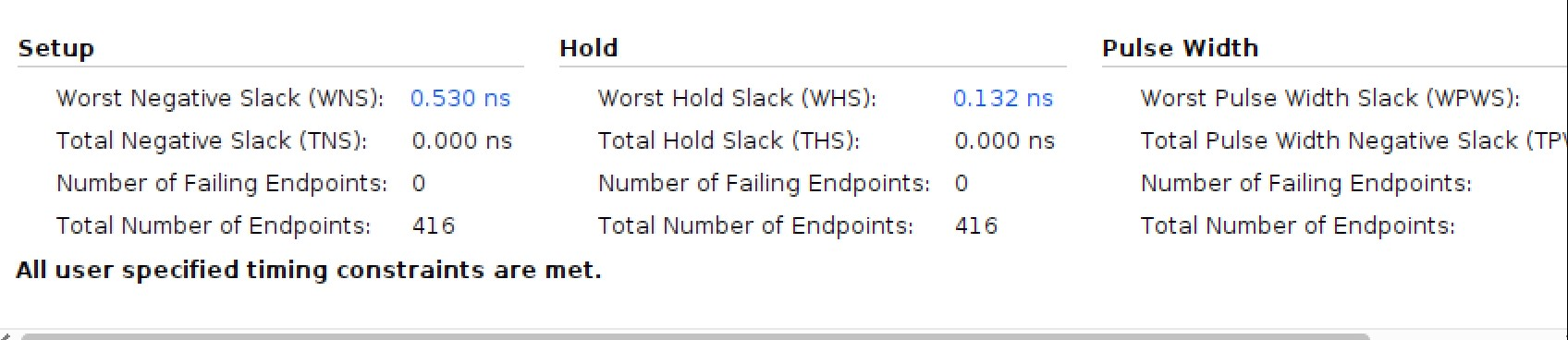
Improved:921775ns

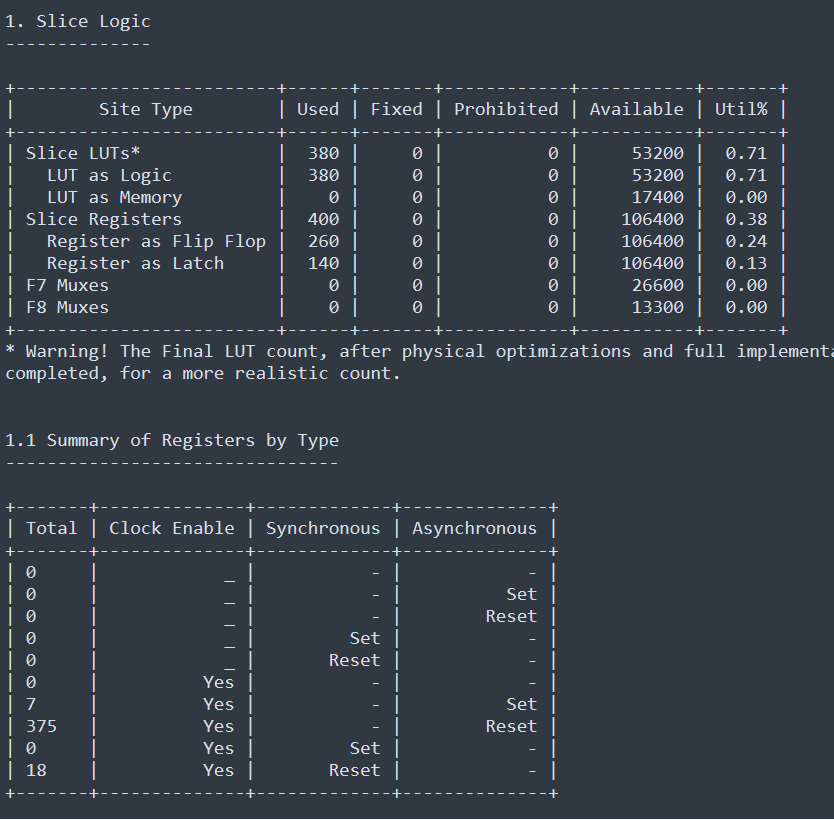


控制feedback 的時間，如下圖

****

* **TIME REPORT**



****

