**SOC Design Laboratory**

**Lab 5 – Caravel SoC FPGA Integration**

Group 6: 112061611 陳伯丞

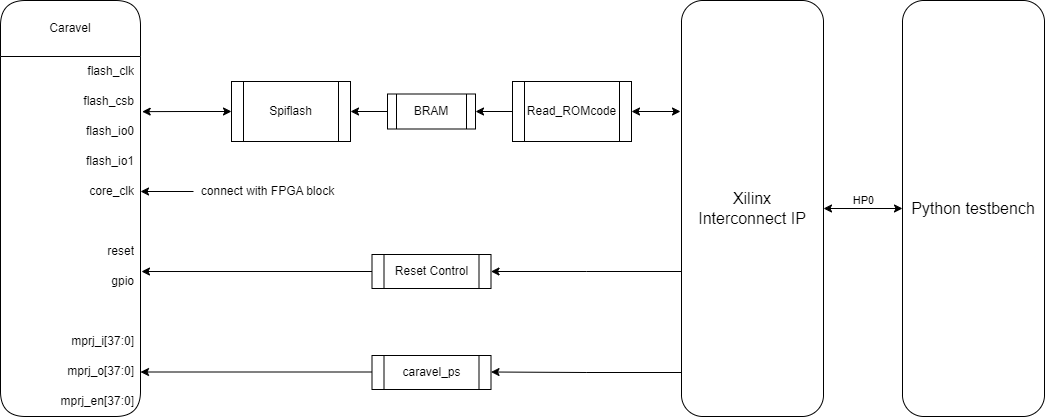
112061524 葉又菘

110063553 張傑閔

**Block Diagram:**

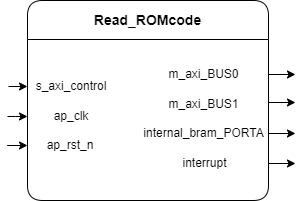
本次lab 5主要為三個部分組成，為Read\_ROMcode、Spiflash、Caravel三部分，下圖是整個lab 5的架構圖：

The lab5 architecture:

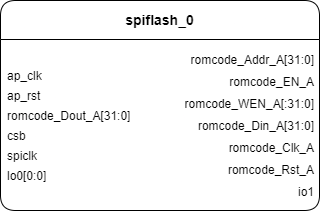


並且以下展示三個部分的block diagram。

Read\_ROMcode block diagram:

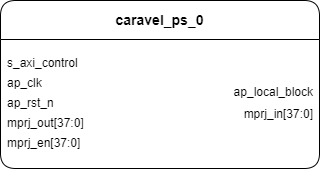


Spiflash block diagram:

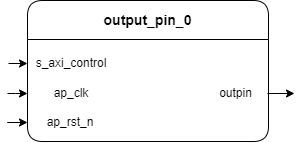


Caravel block diagram:

Caravel PS:

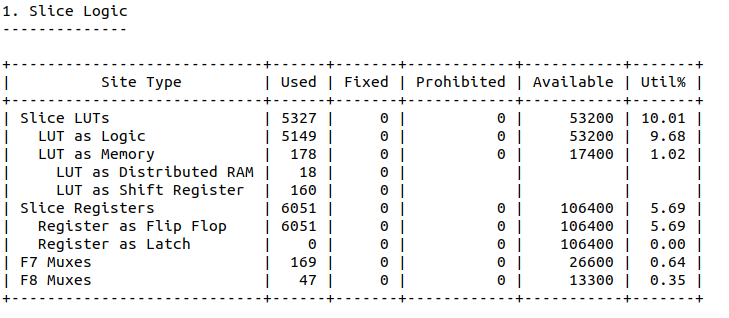


Caravel reset control:

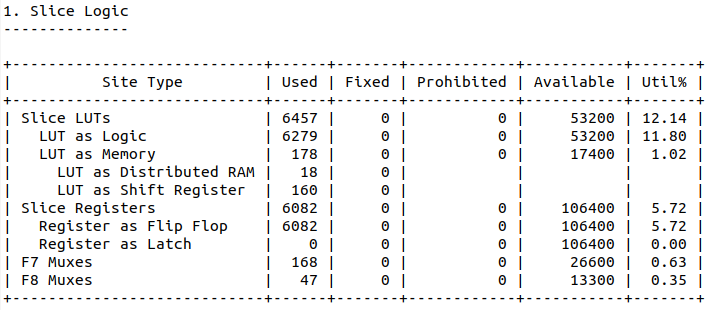


**FPGA utilization:**

Counter utilization:



GCD utilization:



**Explain the function of IP:**

HLS IP read\_romcode:

此IP是由read\_romcode.cpp中設計，主要的功能為使用AXI-Master寫入PS Memory中，能夠透過AXI-Master將caravel testbench轉換成的RISCV code (in hex file)載入至PS Memory buffer中。再將host端設計的code載入至BRAM之中，最後將兩者buffer中的content進行比較是否相同。

Control flow:

讀取DRAM中的ROM code，PS透過m\_axi\_BUS0給予address並且要求讀取，然後此IP就會讀取DRAM ROM code並且回傳PS；相反的，寫入ROM code時，PS透過m\_axi\_BUS1給予address並且要求寫入，然後此IP就會寫入DRAM ROM code並且回傳PS。

HLS IP caravel\_ps:

此IP是由caravel\_ps.cpp中設計，主要的功能為提供PS CPU一個AXI-Lite interface，因此PS CPU 能夠讀取MPRJ\_IO/OUT/EN bits。

HLS IP ResetControl:

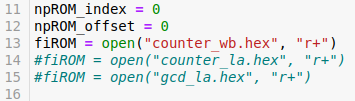
此IP是由output\_pin.cpp中設計，主要的功能為輸出1或0以開啟或關閉Caravel reset pin。並且在提供PS CPU 與上述不同的一個AXI-Lite interface，因此PS CPU能夠控制output。

spiflash

此部分為BRAM與Caravel之間的SPI interface control。用以暫存BRAM輸出資料的flash記憶體，並當作SPI slave進行資料傳輸，在CPU讀取BRAM資料時將資料傳輸給CPU。

**FPGA running result**

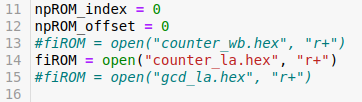
In the first run, select file.

****

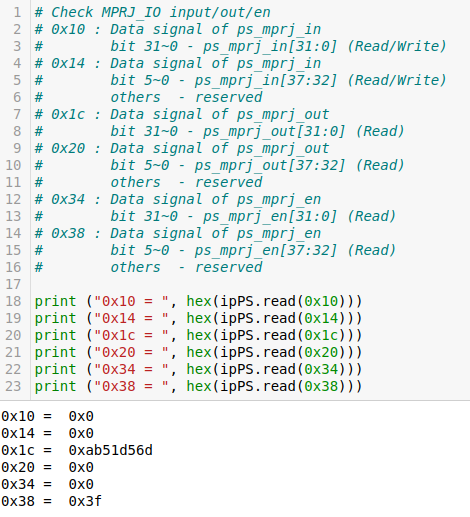
And the result is the following, the value at address 0x1c is 0xab61.



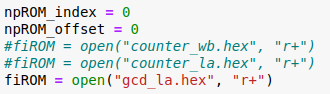
In the second run, we select the file.



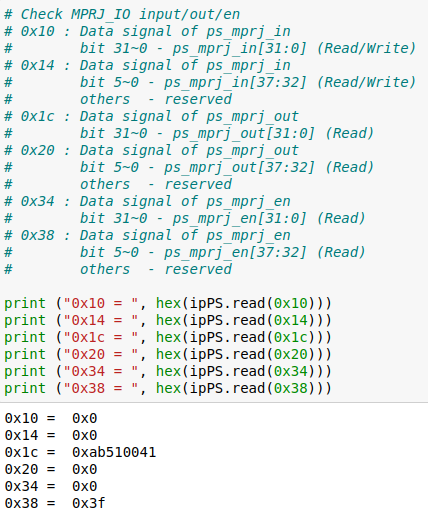
The value at address 0x1c is 0xab51.



Last, select the file.

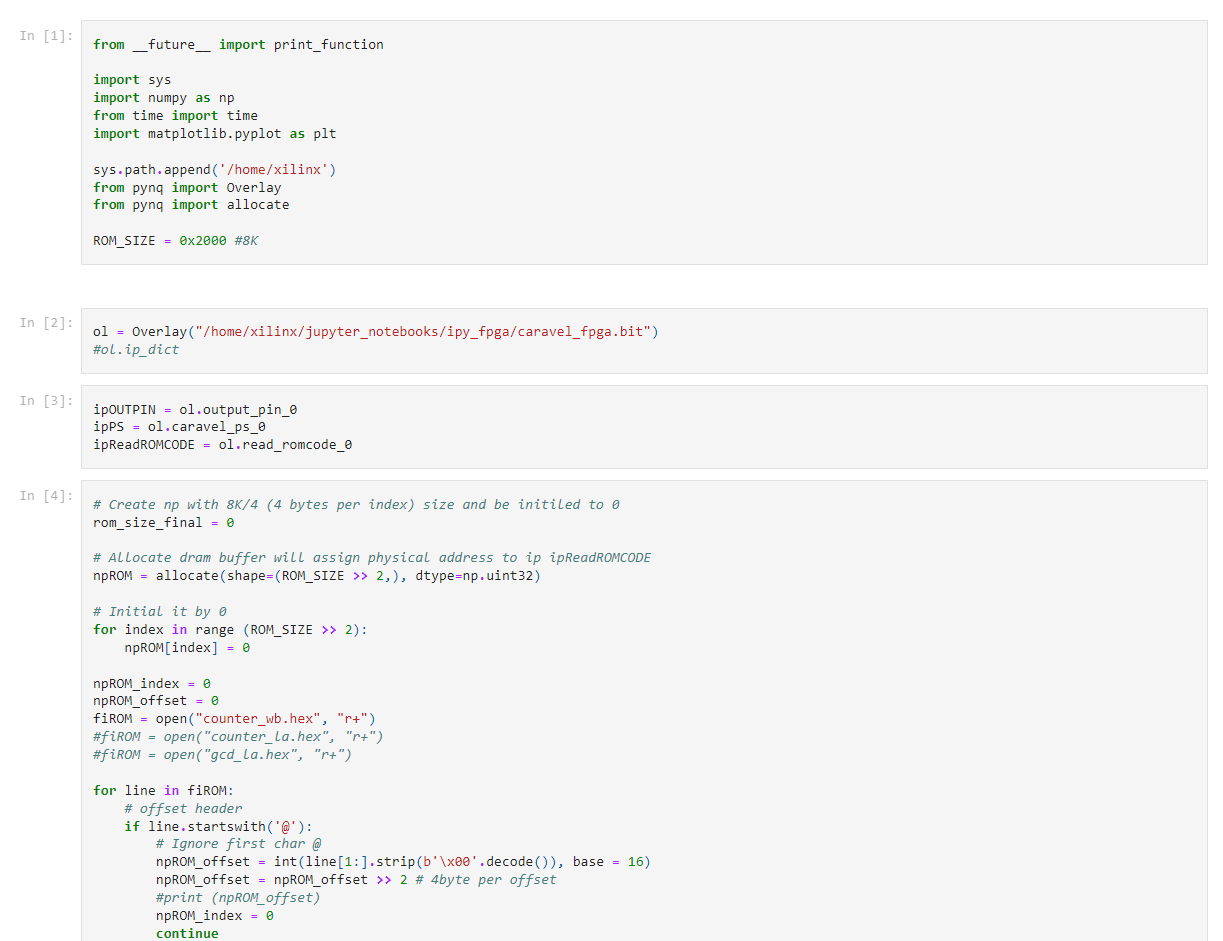


The value at address 0x1c is 0xab51.



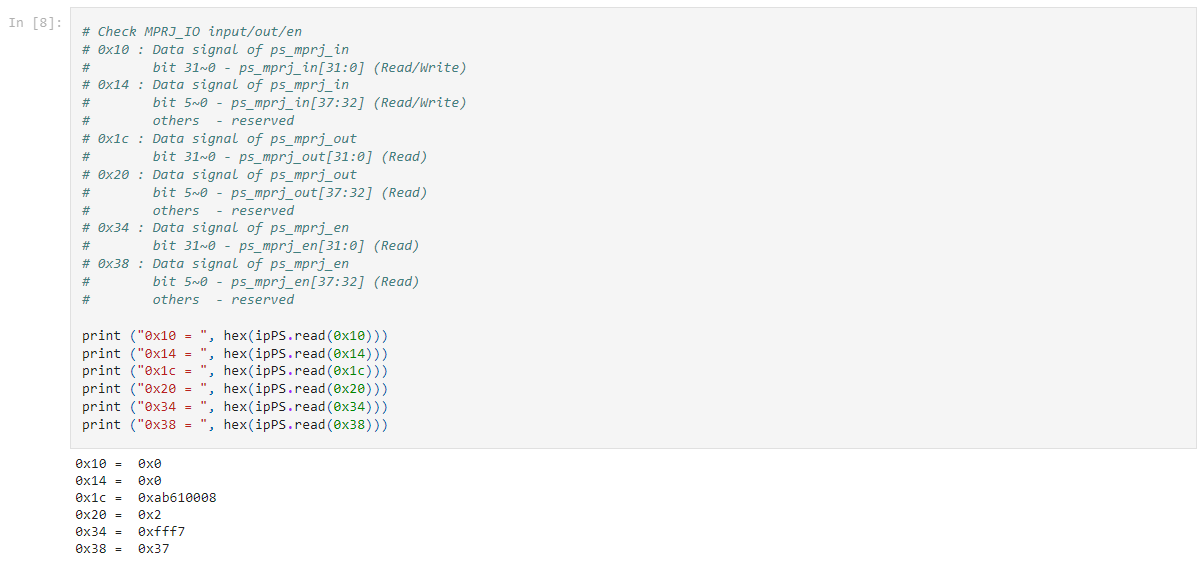
**Results on Jupyter**

“counter\_wb.hex”:

****

****

****

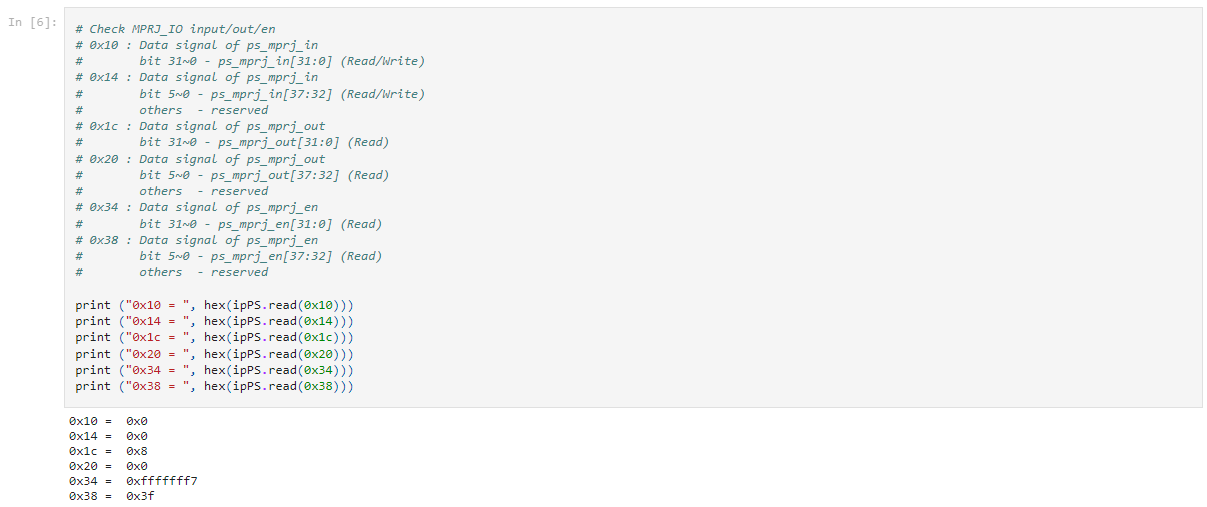
****

“counter\_la.hex”:



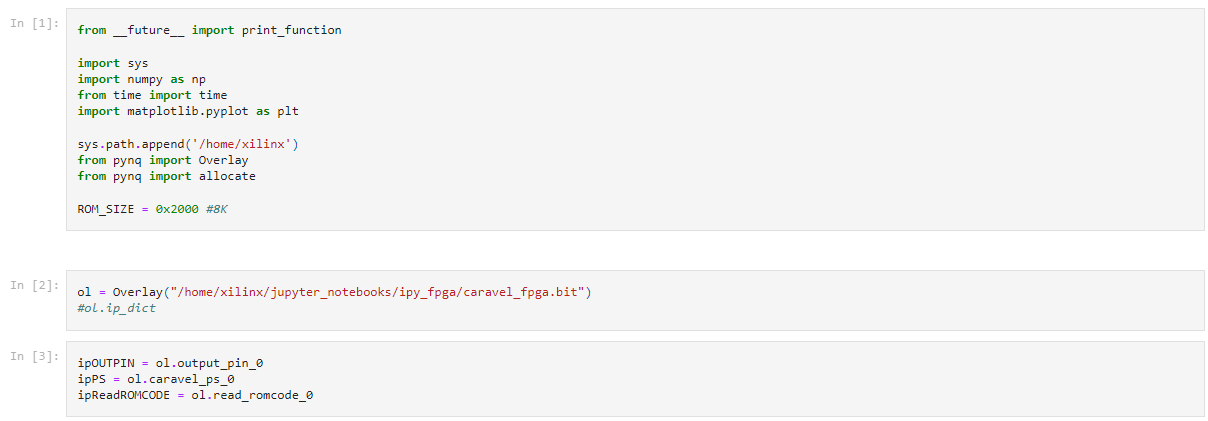


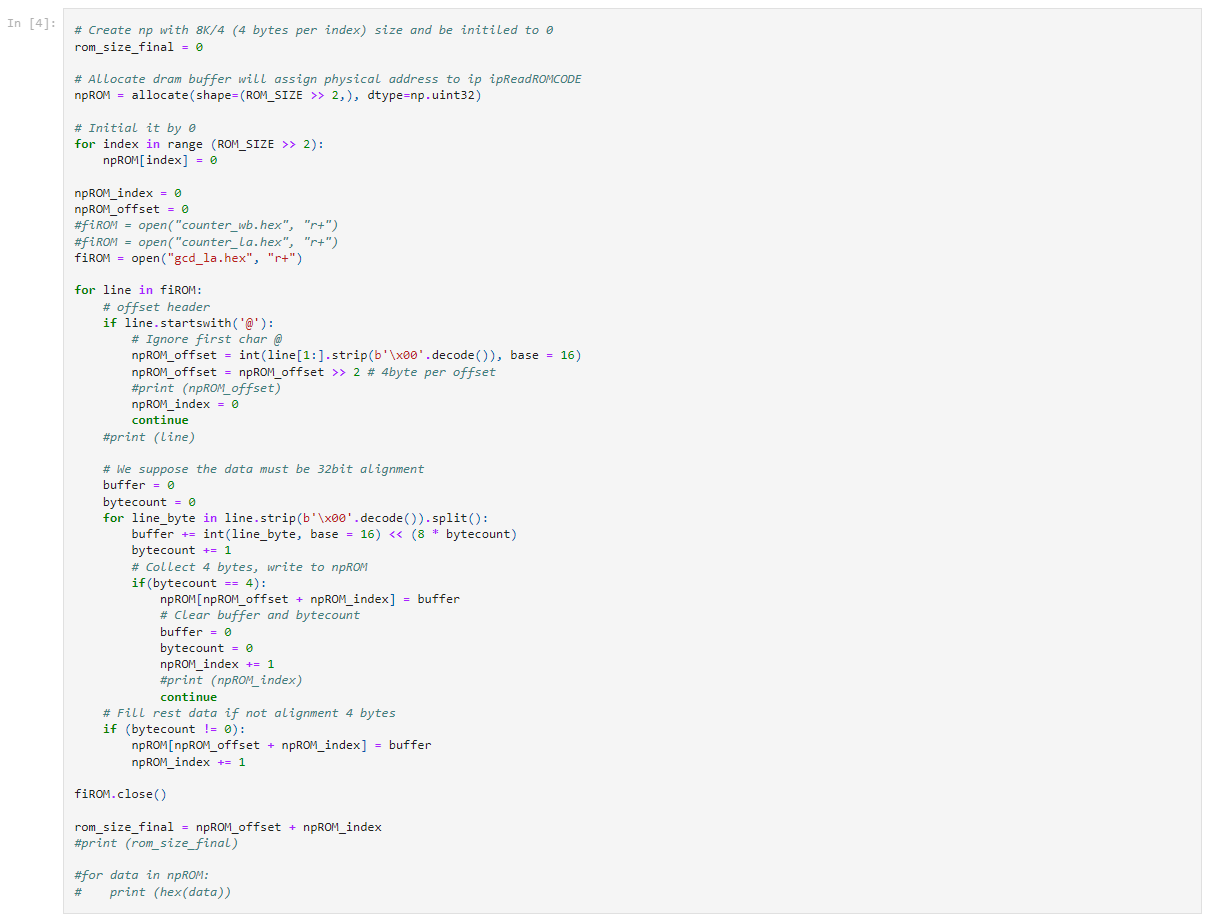


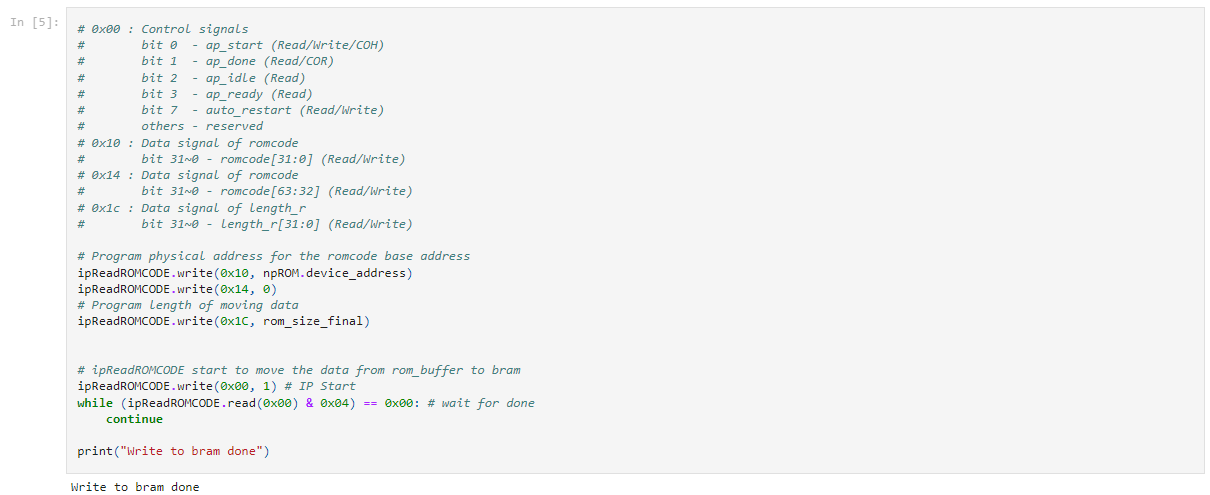


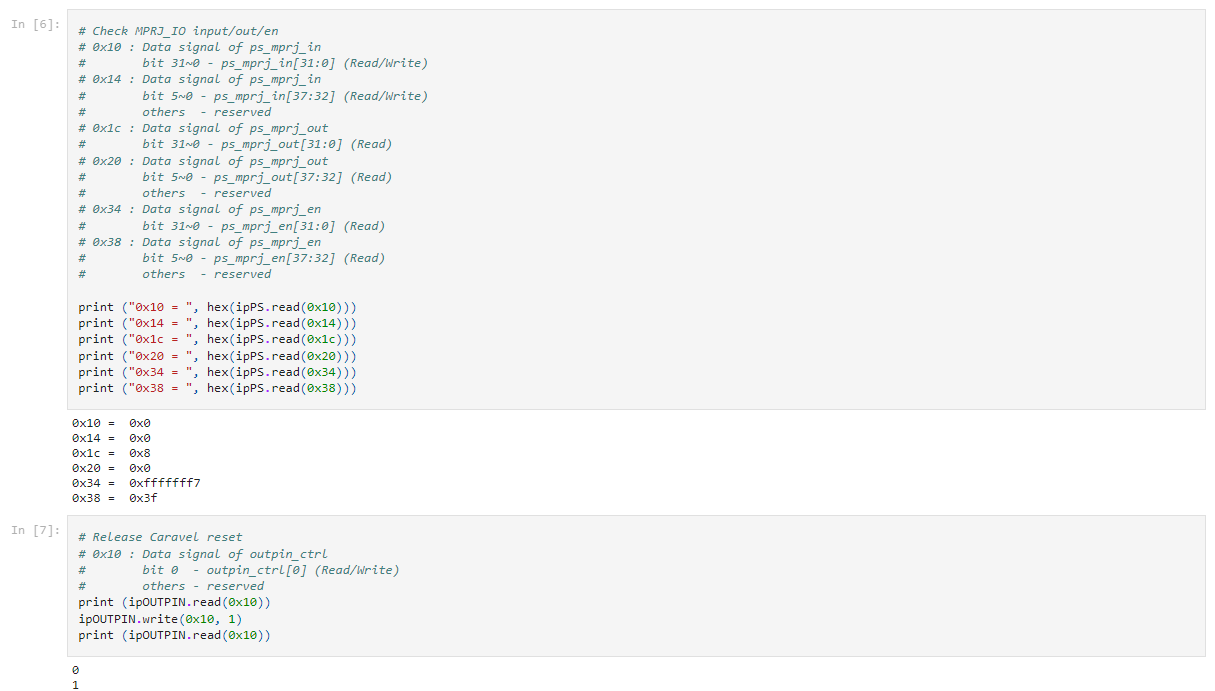


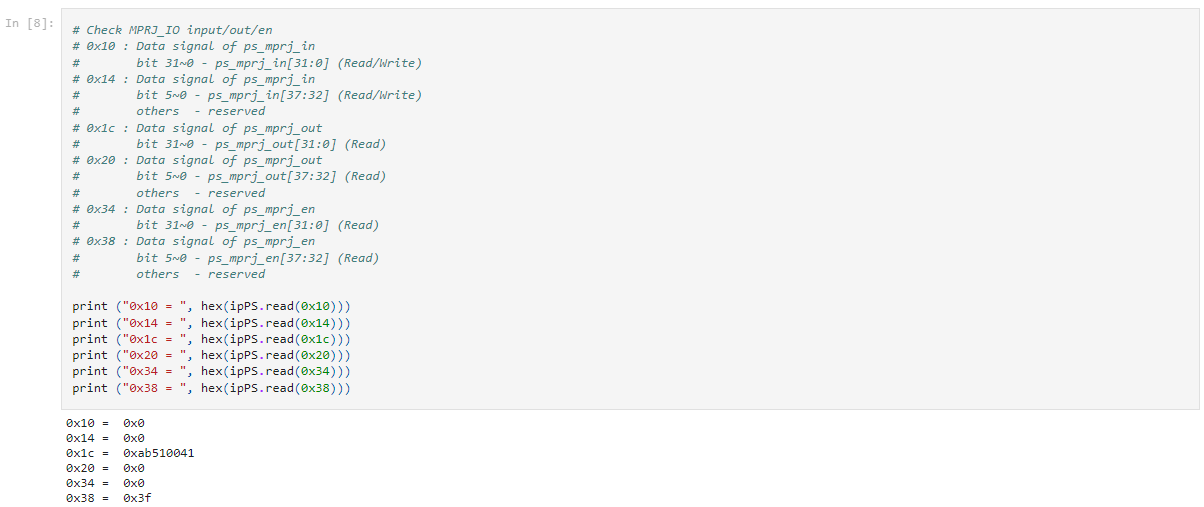
“gcd\_la.hex”:











**Study caravel\_fpga.ipynb, and be familiar with caravel SoC control flow**

本次lab 5我們需要實作在FPGA版上，因此租借onlineFPGA board進行實作。而我們發現使用python notebook的優點有許多，其中最重要的優點就是notebook支援多種的programming language，而我們這次lab 5需要使用PYNQ-Z2 board就是使用python language，因此非常方便。我們可以將所需的bit hwh hex file上傳至python notebook kernel，即可在PYNQ-Z2上進行實作。

以下是在notebook中學習到的實作流程說明。

首先在第二個cell中我們先讀取新的bitstream file，使用函數overlay(“file name”) 讀取設計好的bitstream，並且同時在第三個cell之中把bitstream file裡面的三個設計好的IP讀取出來。

第二步，在FPGA板上宣告一個DRAM buffer，使用allocate(“rom size”)宣告。並且將對應的hex file (firmware c code to RISCV code)打開並讀取。在hex file裡，檔案的開始是@為記號，並且其中儲存的內容是每四個bytes為一單位，因此我們將hex file裡面的資料每4 bytes寫入至剛剛宣告的DRAM buffer，也就是bytecount==4時將資料寫入buffer。

第三步，寫入DRAM buffer後需要reset caravel CPU，因此caravel reset control IP會輸出GPIO進行reset。經過reset之後caravel CPU即可開始透過SPI interface (上述的SPI flash)進行DRAM buffer中儲存的code讀取並執行。

最後，Caravel CPU輸出的結果會經由MPRJ\_I/O/EN輸出，使用caravel ps IP進行讀取並且將讀取結果印在notebook上，並且最終結果會顯示在0x1c的地址的前16個bits之中，比較其結果是否與firmware結果相同即完成。

**Reference**

Course material

TA lecture & PPT