**國立清華大學**

**系統晶片設計**

**SOC Design**

一張含有 文字, 筆跡, 字型, 圓形 的圖片

自動產生的描述

**Lab 5**

**組別: 第12組**

**學號:111063548、111061624、112501538**

**姓名:蕭方凱、尤弘瑋、葉承泓**

**指導老師:賴瑾教授**

**目錄**

[1. Block Diagram 3](#_Toc151935081)

[2. FPGA Utilization 4](#_Toc151935082)

[(1). Caravel utilization 4](#_Toc151935083)

[(2). Read\_romcode utilization 5](#_Toc151935084)

[(3). Caravel\_ps utilization 6](#_Toc151935085)

[(4). Output\_pin utilization 7](#_Toc151935086)

[(5). Utilization總整理（counter） 7](#_Toc151935087)

[3. Explain the function of IP in this design 8](#_Toc151935088)

[(1). read\_romcode 8](#_Toc151935089)

[(2). spiflash 9](#_Toc151935090)

[(3). ResetControl 10](#_Toc151935091)

[(4). caravel\_ps 11](#_Toc151935092)

[4. Run these workload on caravel FPGA 12](#_Toc151935093)

[counter\_wb.hex 12](#_Toc151935094)

[counter\_la.hex 13](#_Toc151935095)

[gcd\_la.hex 14](#_Toc151935096)

[5. Screenshot of Execution result on all workload 15](#_Toc151935097)

[Counter\_wb 15](#_Toc151935098)

[Counter\_la 16](#_Toc151935099)

[Gcd 17](#_Toc151935100)

[6. Study caravel\_fpga.ipynb, and be familiar with caravel SoC control flow 18](#_Toc151935101)

# Block Diagram

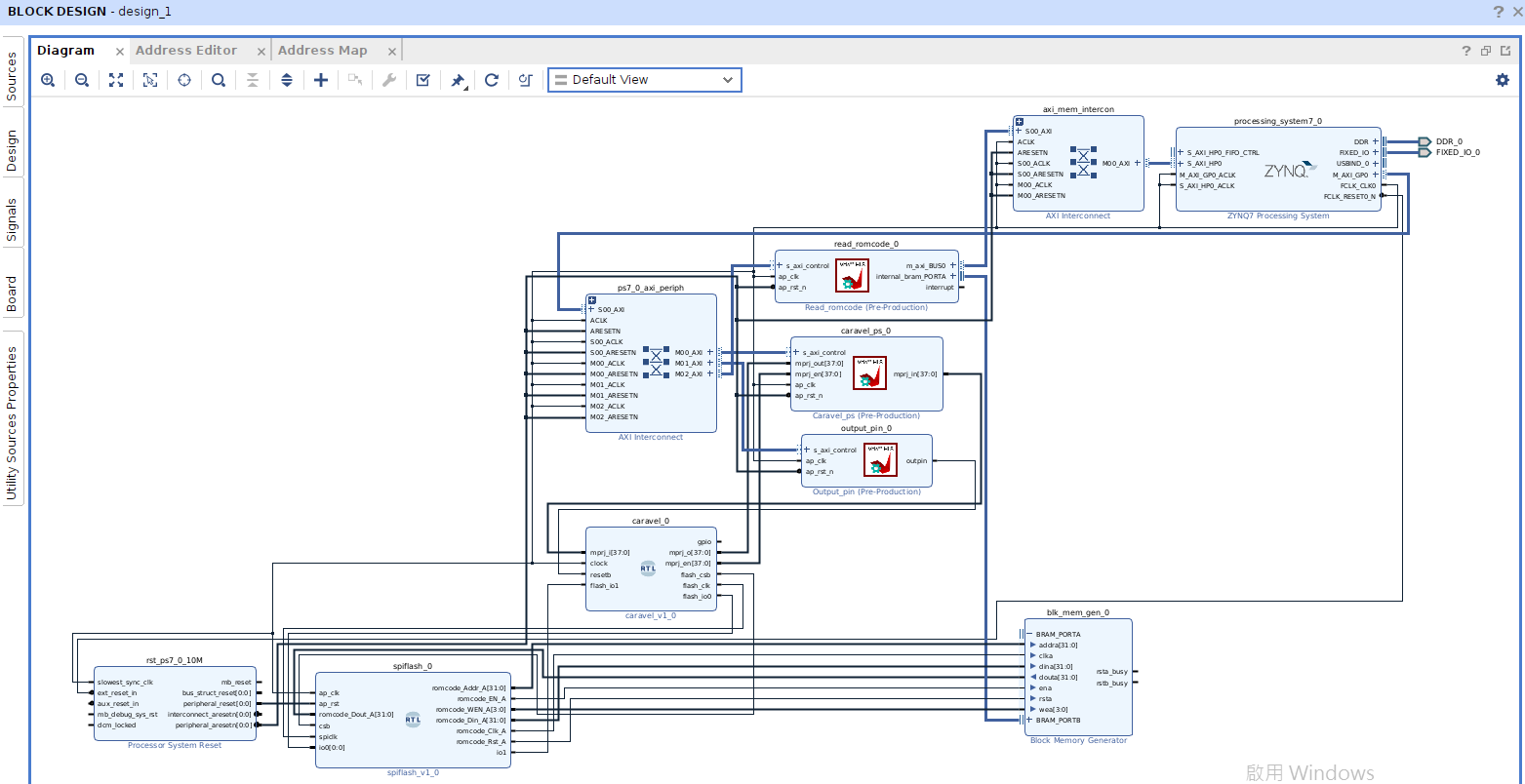


Fig 1 block diagram

# FPGA Utilization

## Caravel utilization

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

## Read\_romcode utilization

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

## Caravel\_ps utilization

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

## Output\_pin utilization

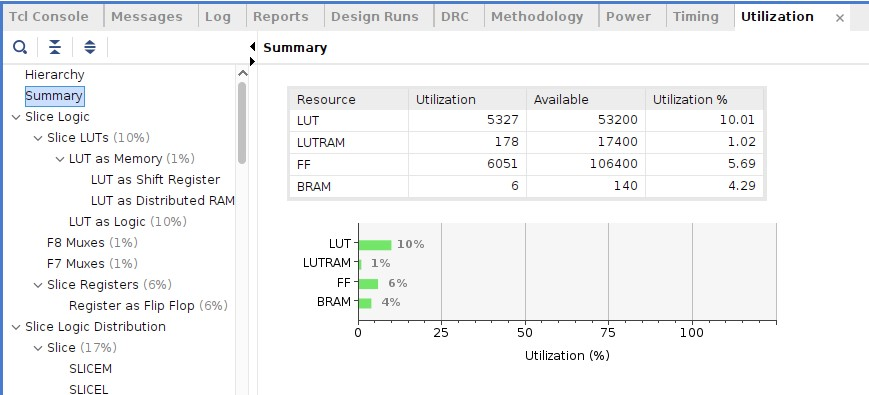
一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

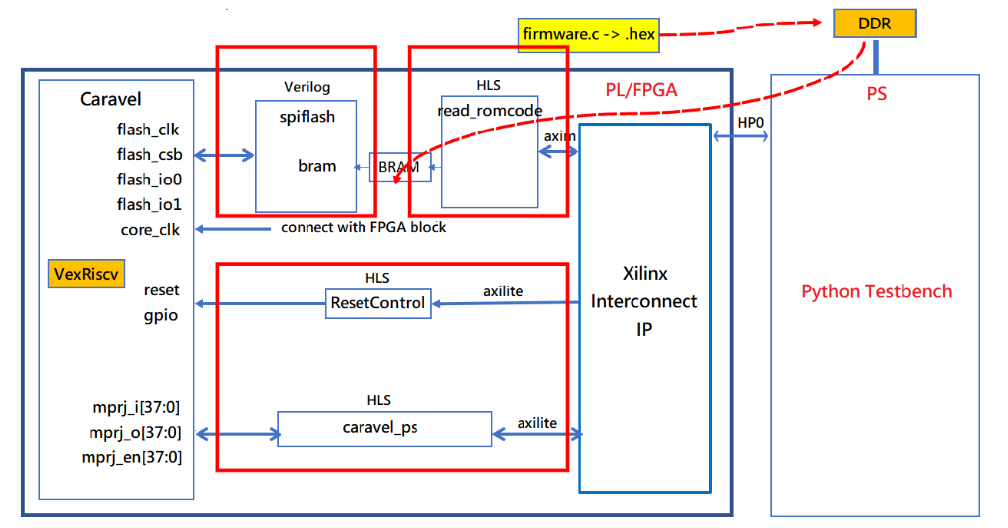
自動產生的描述

## Utilization總整理（counter）



# Explain the function of IP in this design

在底下，我們主要說明的是Github上labi的說明PDF檔（/caravel-soc\_fpga-lab/labi/lab5-caravel FPGA.pdf）中下圖中有框起來的這4個module的功能：



## read\_romcode

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

Fig 2 read\_romcode.cpp

在執行run\_vitis.sh時，會透過Vitis\_HLS將/labi/vitis\_hls\_project/hls\_read\_romcode /src/read\_romcode.cpp進行高階合成產生IP並export成Vivado軟體的相容格式。因此，read\_ROMcode這個module的主要功能及行為定義在read\_romcode.cpp中，如上圖所示。其中的romcode[]這個array主要存放software code經過compile而得到的firmware code（binary code），是由PS side的DDR memory input而來的，而internal\_bram[]這個array則是對應到與BRAM之間的接口（interface），這個interface之間的protocol是使用”bram”的interface，因此圖中使用

*#pragma HLS INTERFACE bram port=internal\_bram*

這個#pragma來限制合成方式。上述兩個array的大小皆被限制為8KB，因此只能存放”8KB/sizeof(int)”個整數。

圖中的”length”則是code的大小，也就是binary file（即compile後產生的.hex檔）中以整數為單位的長度，這個資訊也會由PS side來提供，interface是使用AXI-Lite來提供資訊。由於array的大小被限制在8KB以內，因此當code size大過這個大小時（也就是圖中 ”if(length > (CODE\_SIZE/sizeof(int)))” 條件判斷式），就只能領到8KB的大小，因此length就會被限制在” CODE\_SIZE/sizeof(int)”。

接者就是此module最主要的功能：將ROM code存放至BRAM中，也就是將這些code一行行放入與BRAM之間的interface讓BRAM去存取，因此使用

*for(i = 0; i < length; i++) {*

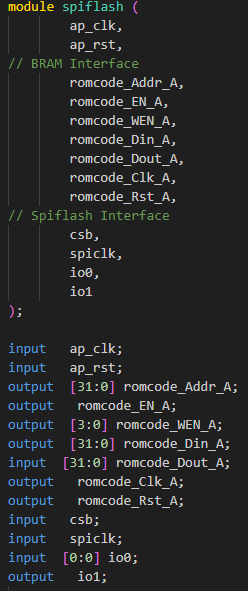
*internal\_bram[i] = romcode[i];*

*}*

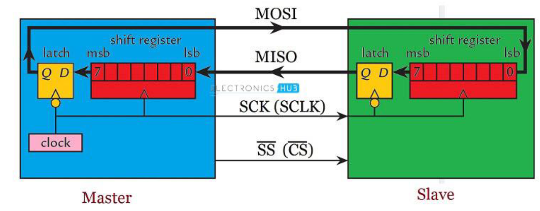
來達成，並使用pipeline的#pragma來加速存取。

## spiflash

spiflash位於/caravel-soc\_fpga-lab/labi/vvd\_srcs/spiflash.v，為Caravel SoC要向BRAM索取CPU要執行的firmware code時的橋樑，因此它有兩個interface — BRAM以及Caravel SoC (spiflash)，如下圖所示：



由於BRAM中的code為read-only，因此romcode\_WEN\_A一直為0，而要讀取BRAM的address則是依照spi\_address來提供，BRAM輸出的Data\_out (romcode\_Dout\_A) 則依照address（byte-address的形式）取出並存到memory[7:0]中，共存了1個byte起來。接著按照下圖的方式輸出及輸入：



上圖為labi的說明PDF檔（/caravel-soc\_fpga-lab/labi/lab5-caravel FPGA.pdf）中的介紹圖。Spi input進來的io0這個bit會被放到buffer的末端，形成shift register，buffer會依據bytecount（每個cycle bitcount會增加1，而當bitcount滿7後，就會trigger bytecount增加1）而決定要放到spi\_addr的哪一個byte位置。由於只支援spi\_cmd == 'h03（即read command），BRAM會依據address依序吐出ROMcode，並且再透過shift register的方式將memory[7:0]的值依序放至outbuf中，再一個bit接著一個bit輸出到io1（即outbuf[7]位置）中輸出給Caravel SoC。

## ResetControl

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

Fig 3 output\_pin.cpp

在執行run\_vitis.sh時，會透過Vitis\_HLS將/labi/vitis\_hls\_project/hls\_output\_pin /src/output\_pin.cpp進行高階合成產生IP並export成Vivado軟體的相容格式。因此，ResetControl這個module的主要功能及行為定義在output\_pin.cpp中，如上圖所示。此module的主要功能用於將一個布林值 outpin\_ctrl 複製給另一個布林值引用outpin。

“outpin = outpin\_ctrl”這一行的作用是將 outpin\_ctrl 的值賦給 outpin，這樣 outpin 就會擁有和 outpin\_ctrl 相同的布林值。

## caravel\_ps

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

Fig 4 caravel\_ps.cpp

在執行run\_vitis.sh時，會透過Vitis\_HLS將/labi/vitis\_hls\_project/hls\_caravel\_ps /src/caravel\_ps.cpp進行高階合成產生IP並export成Vivado軟體的相容格式。因此，caravel\_ps這個module的主要功能及行為定義在caravel\_ps.cpp中，如上圖所示。此module的主要功能為提供PS CPU AXI Lite 介面以讀取MPRJ\_IO/OUT/EN bits，並透過HLS 實作並匯出IP以供Vivado 專案使用。

PS 端的輸入和輸出接口: ps\_mprj\_in、ps\_mprj\_out、ps\_mprj\_en

Caravel flash端的輸入和輸出接口: mprj\_in、mprj\_out、mprj\_en

函數將值從 Caravel 閃存端（mprj\_out 和 mprj\_en）複製到 PS 端（ps\_mprj\_out 和 ps\_mprj\_en）。如果 mprj\_en[i] 為真，則將 mprj\_out[i] 複製到 mprj\_in[i]；否則，將 ps\_mprj\_in[i] 複製到 mprj\_in[i]。#pragma HLS UNROLL 指令表明這個循環可以展開以提高性能。

# Run these workload on caravel FPGA

## counter\_wb.hex

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

## counter\_la.hex

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

## gcd\_la.hex

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

# Screenshot of Execution result on all workload

## Counter\_wb

一張含有 文字, 螢幕擷取畫面, 軟體, 陳列 的圖片

自動產生的描述

最終0x1c的位置的值最高bit為AB61，符合期待值！

## Counter\_la

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

最終0x1c的位置的值最高bit為AB51，符合期待值！

## Gcd

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

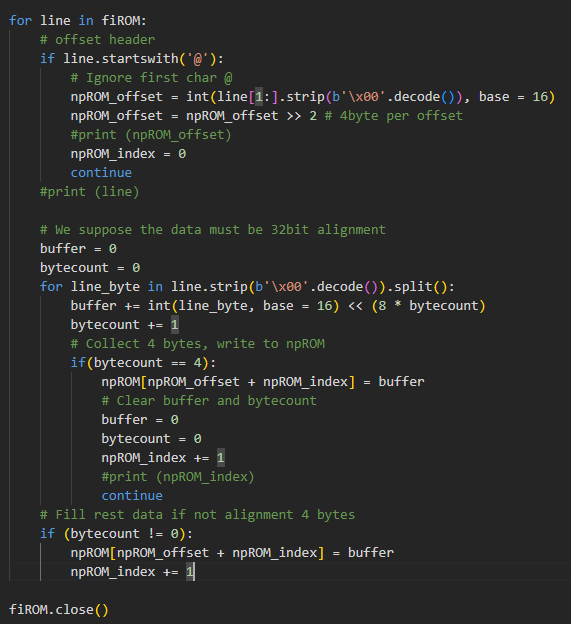
一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

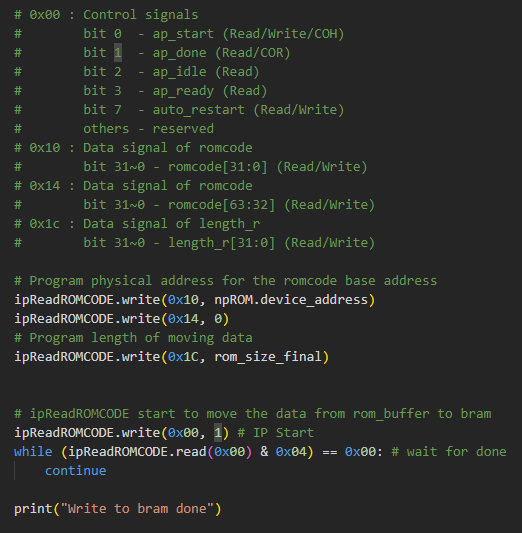
最終0x1c的位置的值最高bit為AB40，符合期待值！

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

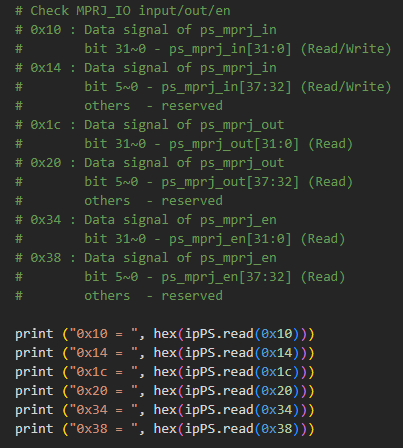
Step 1. 將hex檔讀進ROM裡



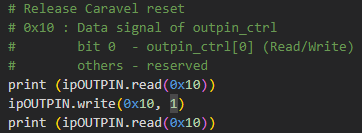
Step 2. 將ROM code放進BRAM



Step 3. 讀取mprj\_in的值



Step 4. RESET



Step 5. 讀取mprj\_out的值

