SOC Design

Lab #5 - Caravel FPGA

Group no: 5

Members:

M11207415 陳謝鎧

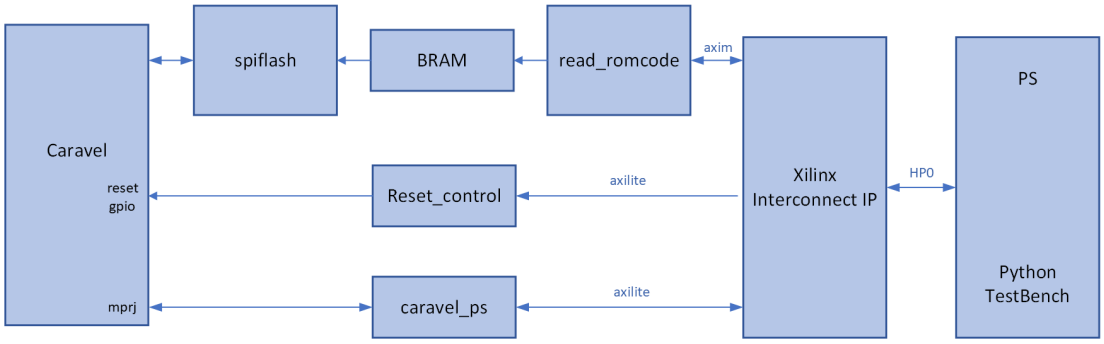
M11207002 陳泊佑

M11107426 廖千慧

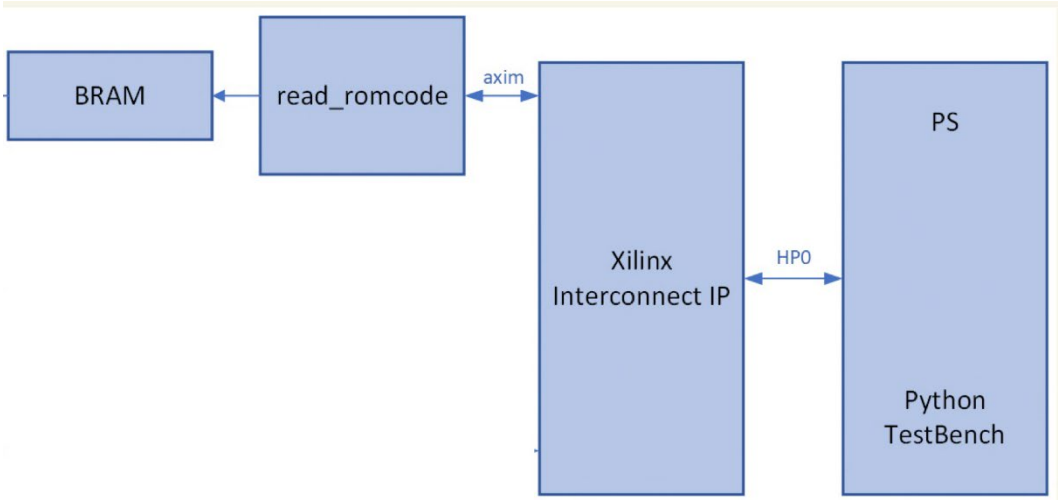
M11207328 吳奕帆

Block diagram:

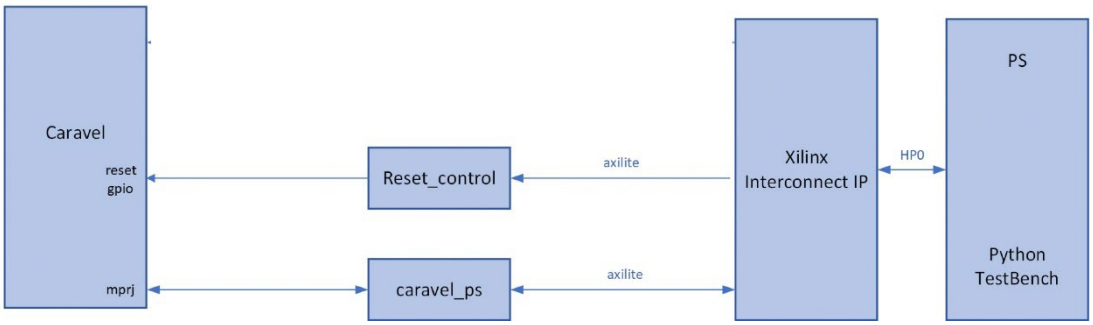
- The hole architecture:



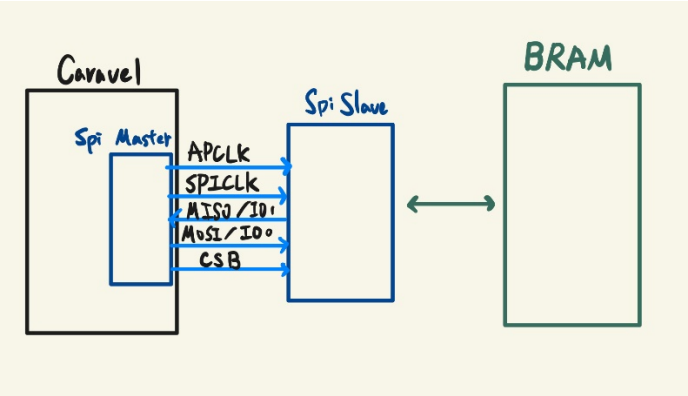
- The part of Read_ROMcode:



- The part of Caravel:



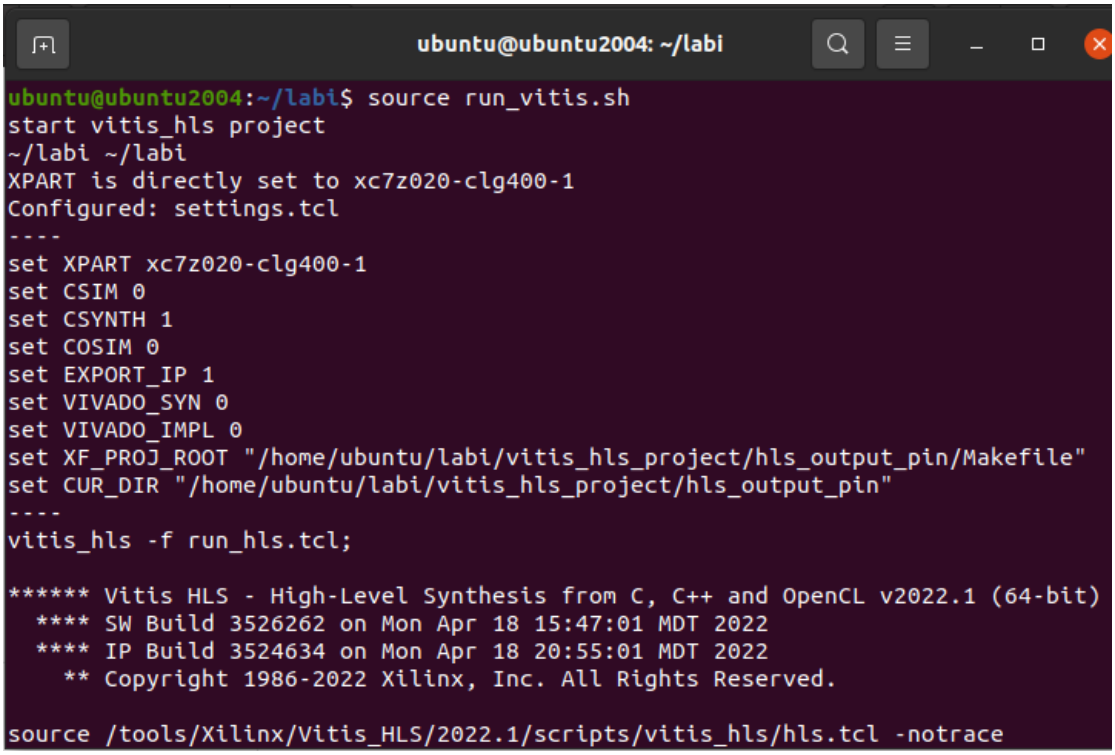
➢ The part of Spiflash IP and SPI interface:

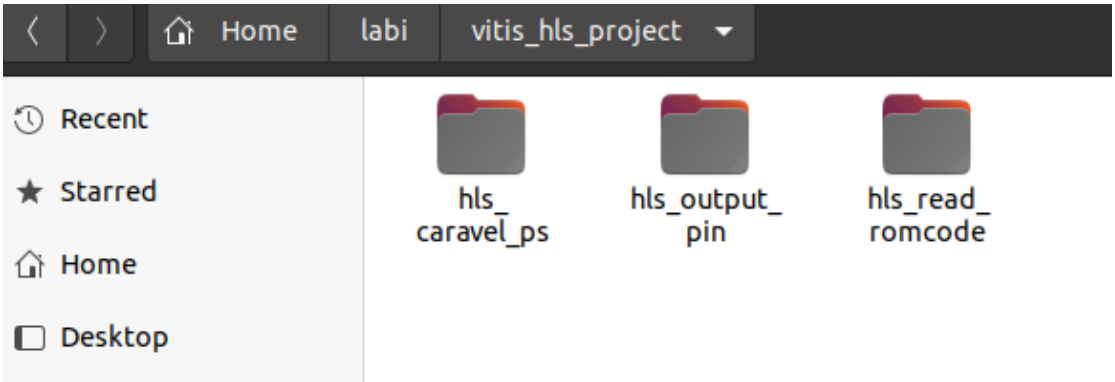


Practical operation:

- Run Vitis:

  執行 source run_vitis.sh，所有的 HLS 專案將會自動構建並導出 IP。





- Run Vivado:

  執行 source run_vivado.sh，生成 clock 50MHzm 用來當作 bitstream of user project counter。

```
ubuntu@ubuntu2004: ~/labi
ubuntu@ubuntu2004:~/labi$ source run_vivado.sh
start vivado project
remove previous project
rm: cannot remove 'timingreport.txt': No such file or directory

****** Vivado v2022.1 (64-bit)
  **** SW Build 3526262 on Mon Apr 18 15:47:01 MDT 2022
  **** IP Build 3524634 on Mon Apr 18 20:55:01 MDT 2022
    ** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

source vvd_caravel_fpga.tcl
# proc checkRequiredFiles { origin_dir} {
#   set status true
#   set files [list \
#  "[file normalize "$origin_dir/vvd_srcs/spiflash.v"]"\
```

若是執行 source run_vivado_gcd.sh，則會生成 clock 10MHzm 用來當作 bitstream of user project counter。

```
ubuntu@ubuntu2004: ~/labi
ubuntu@ubuntu2004:~/labi$ source run_vivado_gcd.sh
start vivado project
remove previous project

****** Vivado v2022.1 (64-bit)
  **** SW Build 3526262 on Mon Apr 18 15:47:01 MDT 2022
  **** IP Build 3524634 on Mon Apr 18 20:55:01 MDT 2022
    ** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

source vvd_caravel_fpga_gcd.tcl
# proc checkRequiredFiles { origin_dir} {
#   set status true
#   set files [list \
#  "[file normalize "$origin_dir/vvd_srcs/spiflash.v"]"\
#  "[file normalize "$origin_dir/vvd_srcs/caravel_soc/vip/RAM128.v"]"\
```

- Online FPGA:

  租借線上 FPGA 版並用 jupyter notebook 執行。

FPGA utilization:

- Run_vivado_utilization:

```
1. Slice Logic
--------------

+----------------------------+------+-------+------------+-----------+-------+
|          Site Type         | Used | Fixed | Prohibited | Available | Util% |
+----------------------------+------+-------+------------+-----------+-------+
| Slice LUTs                 | 5327 |     0 |          0 |     53200 | 10.01 |
|   LUT as Logic             | 5149 |     0 |          0 |     53200 |  9.68 |
|   LUT as Memory            |  178 |     0 |          0 |     17400 |  1.02 |
|     LUT as Distributed RAM |   18 |     0 |            |           |       |
|     LUT as Shift Register  |  160 |     0 |            |           |       |
| Slice Registers            | 6051 |     0 |          0 |    106400 |  5.69 |
|   Register as Flip Flop    | 6051 |     0 |          0 |    106400 |  5.69 |
|   Register as Latch        |    0 |     0 |          0 |    106400 |  0.00 |
| F7 Muxes                   |  169 |     0 |          0 |     26600 |  0.64 |
| F8 Muxes                   |   47 |     0 |          0 |     13300 |  0.35 |
+----------------------------+------+-------+------------+-----------+-------+
```

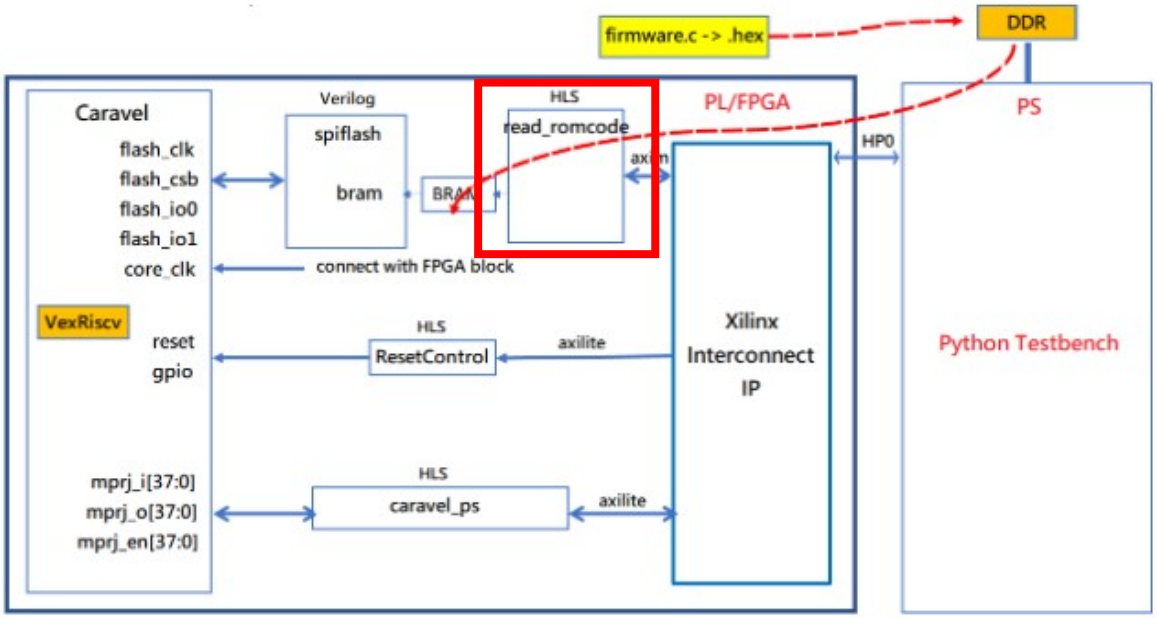
- Run_vivado_gcd_utilization:

```
1. Slice Logic
--------------

+----------------------------+------+-------+------------+-----------+-------+
|          Site Type         | Used | Fixed | Prohibited | Available | Util% |
+----------------------------+------+-------+------------+-----------+-------+
| Slice LUTs                 | 6457 |     0 |          0 |     53200 | 12.14 |
|   LUT as Logic             | 6279 |     0 |          0 |     53200 | 11.80 |
|   LUT as Memory            |  178 |     0 |          0 |     17400 |  1.02 |
|     LUT as Distributed RAM |   18 |     0 |            |           |       |
|     LUT as Shift Register  |  160 |     0 |            |           |       |
| Slice Registers            | 6082 |     0 |          0 |    106400 |  5.72 |
|   Register as Flip Flop    | 6082 |     0 |          0 |    106400 |  5.72 |
|   Register as Latch        |    0 |     0 |          0 |    106400 |  0.00 |
| F7 Muxes                   |  168 |     0 |          0 |     26600 |  0.63 |
| F8 Muxes                   |   47 |     0 |          0 |     13300 |  0.35 |
+----------------------------+------+-------+------------+-----------+-------+
```

Explain the function of IP in this design:

- HLS read_romcode:



透過 read_romcode 可以傳遞資料，將 program(.hex)從 PS 的 DDR 載入到 BRAM。步驟如下:

Step1.增加另一個 AXI-Master 路徑以寫入 PS Memory

Step2.載入 program.hex 從 Caravel 中移至 PS memory buffer

Step3.Develop host code 將 program.hex 載入到 BRAM，並從 BRAM 讀取。

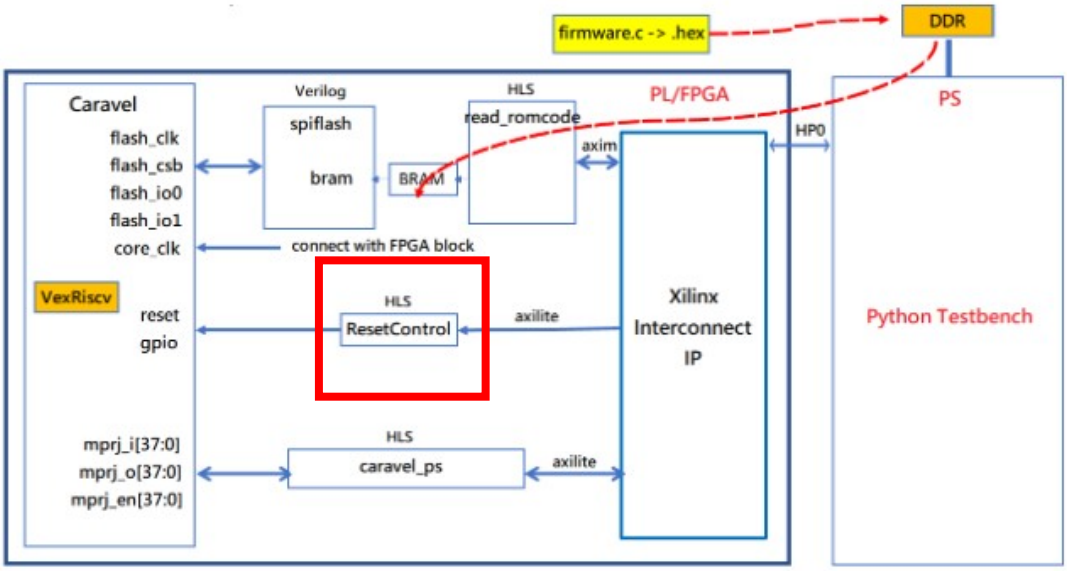
Step4.比較 input buffer 和 out buffer 的內容是否相同。

- Read_ROMcode – control flow

  IP write ROM code to DRAM:

  IP write ROM code to DRAM:

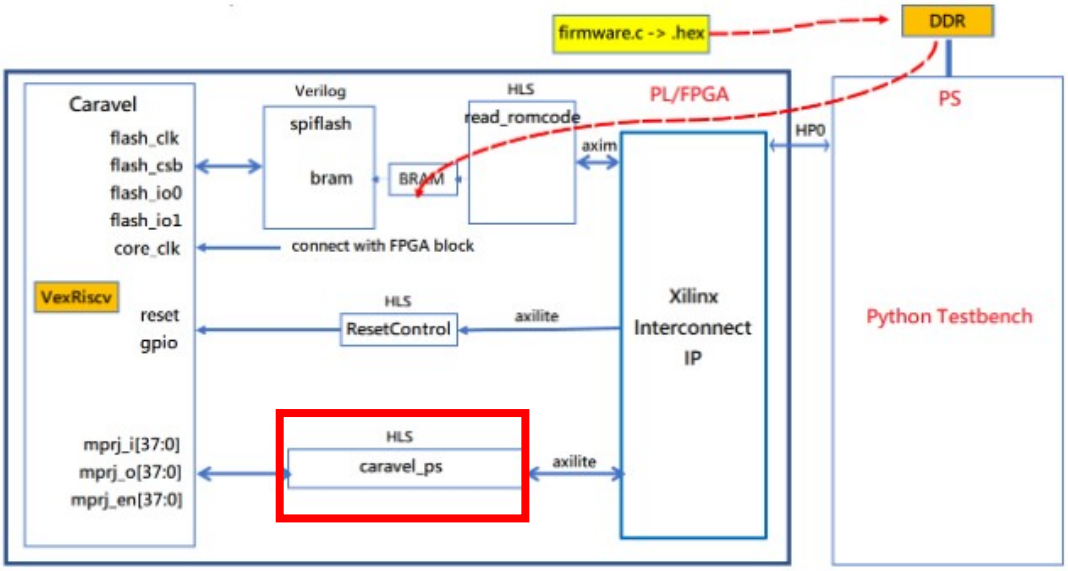
  ➢ HLS ResetControl:



透過 ResetControl ip 設定 reset 信號控制，1 或 0 分別控制 Caravel 的 reset pin 是 assert 還是 de-assert，提供 AXI LITE 用以 PS CPU 的輸出控制。使用 HLS 實現並導

出 IP 以供 Vivado 使用。

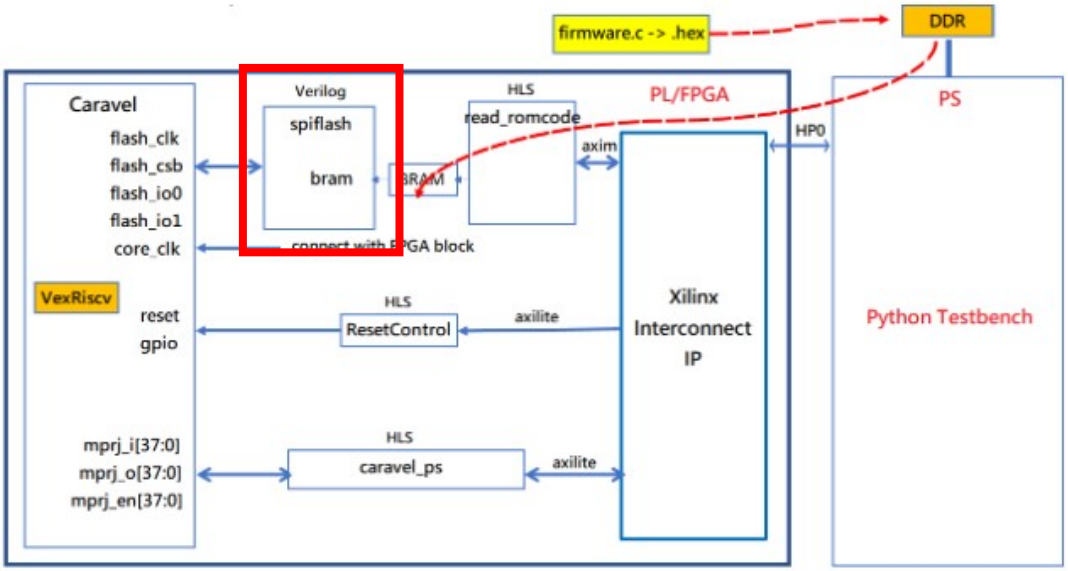
- HLS caravel_ps:



透過 MPRJ_IO 經由 caravel_ps 這個 ip 在執行完後溝通。

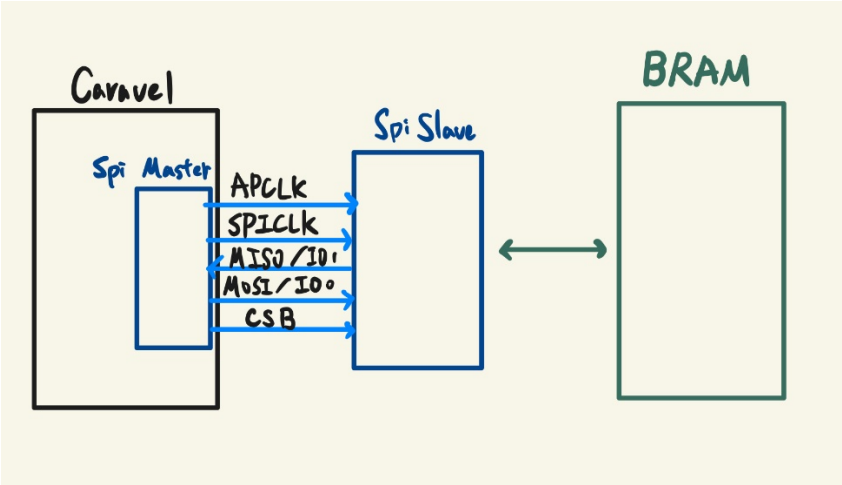
提供 AXI Lite 接口供 PS CPU 讀取 MPRJ_IO/OUT/EN bit，用 HLS 實現並導出 IP 以供 Vivado 使用。

- Verilog Spiflash:



將資料從 BRAM 傳回 Caravel。

SPI 的 interface



Screenshot of Execution result on all workload:

- Execute “counter_wb.hex”

```
In [4]: # Create np with 8K/4 (4 bytes per index) size and be initiled to 0
        rom_size_final = 0

        # Allocate dram buffer will assign physical address to ip ipReadROMCODE
        npROM = allocate(shape=(ROM_SIZE >> 2,), dtype=np.uint32)

        # Initial it by 0
        for index in range (ROM_SIZE >> 2):
            npROM[index] = 0

        npROM_index = 0
        npROM_offset = 0
        fiROM = open("counter_wb.hex", "r+")
        #fiROM = open("counter_la.hex", "r+")
        #fiROM = open("gcd_la.hex", "r+")
```

由執行結果得知，在 0x1C 的位置上值為 0xab61。

```
In [8]: # Check MPRJ_IO input/out/en
        # 0x10 : Data signal of ps_mprj_in
        #        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
        # 0x14 : Data signal of ps_mprj_in
        #        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
        #        others  - reserved
        # 0x1c : Data signal of ps_mprj_out
        #        bit 31~0 - ps_mprj_out[31:0] (Read)
        # 0x20 : Data signal of ps_mprj_out
        #        bit 5~0 - ps_mprj_out[37:32] (Read)
        #        others  - reserved
        # 0x34 : Data signal of ps_mprj_en
        #        bit 31~0 - ps_mprj_en[31:0] (Read)
        # 0x38 : Data signal of ps_mprj_en
        #        bit 5~0 - ps_mprj_en[37:32] (Read)
        #        others  - reserved

        print ("0x10 = ", hex(ipPS.read(0x10)))
        print ("0x14 = ", hex(ipPS.read(0x14)))
        print ("0x1c = ", hex(ipPS.read(0x1c)))
        print ("0x20 = ", hex(ipPS.read(0x20)))
        print ("0x34 = ", hex(ipPS.read(0x34)))
        print ("0x38 = ", hex(ipPS.read(0x38)))

        0x10 =  0x0
        0x14 =  0x0
        0x1c =  0xab610008
        0x20 =  0x2
        0x34 =  0xfff7
        0x38 =  0x37
```

- All on jupyter notebook

jupyter caravel_fpga Last Checkpoint: 1 小時前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [1]:
```python
from __future__ import print_function

import sys
import numpy as np
from time import time
import matplotlib.pyplot as plt

sys.path.append('/home/xilinx')
from pynq import Overlay
from pynq import allocate

ROM_SIZE = 0x2000 #8K
```

In [2]:
```python
ol = Overlay("/home/xilinx/jupyter_notebooks/ipy_fpga/caravel_fpga.bit")
#ol.ip_dict
```

In [3]:
```python
ipOUTPIN = ol.output_pin_0
ipPS = ol.caravel_ps_0
ipReadROMCODE = ol.read_romcode_0
```

jupyter caravel_fpga Last Checkpoint: 1 小時前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [4]:
```python
# Create np with 8K/4 (4 bytes per index) size and be initiled to 0
rom_size_final = 0

# Allocate dram buffer will assign physical address to ip ipReadROMCODE
npROM = allocate(shape=(ROM_SIZE >> 2,), dtype=np.uint32)

# Initial it by 0
for index in range (ROM_SIZE >> 2):
    npROM[index] = 0

npROM_index = 0
npROM_offset = 0
fiROM = open("counter_wb.hex", "r+")
#fiROM = open("counter_la.hex", "r+")
#fiROM = open("gcd_la.hex", "r+")

for line in fiROM:
    # offset header
    if line.startswith('@'):
        # Ignore first char @
        npROM_offset = int(line[1:].strip(b'\x00'.decode()), base = 16)
        npROM_offset = npROM_offset >> 2 # 4byte per offset
        #print (npROM_offset)
        npROM_index = 0
        continue
    #print (line)

    # We suppose the data must be 32bit alignment
    buffer = 0
    bytecount = 0
    for line_byte in line.strip(b'\x00'.decode()).split():
        buffer += int(line_byte, base = 16) << (8 * bytecount)
        bytecount += 1
        # Collect 4 bytes, write to npROM
        if(bytecount == 4):
            npROM[npROM_offset + npROM_index] = buffer
            # Clear buffer and bytecount
            buffer = 0
            bytecount = 0
            npROM_index += 1
            #print (npROM_index)
            continue
    # Fill rest data if not alignment 4 bytes
    if (bytecount != 0):
        npROM[npROM_offset + npROM_index] = buffer
        npROM_index += 1
```

jupyter caravel_fpga Last Checkpoint: 1 小時前 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

No kernel Trusted Python 3

Run Code Voilà

```
            #print (npROM_index)
            continue
    # Fill rest data if not alignment 4 bytes
    if (bytecount != 0):
        npROM[npROM_offset + npROM_index] = buffer
        npROM_index += 1

fiROM.close()

rom_size_final = npROM_offset + npROM_index
#print (rom_size_final)

#for data in npROM:
#    print (hex(data))
```

In [5]:

```
# 0x00 : Control signals
#        bit 0  - ap_start (Read/Write/COH)
#        bit 1  - ap_done (Read/COR)
#        bit 2  - ap_idle (Read)
#        bit 3  - ap_ready (Read)
#        bit 7  - auto_restart (Read/Write)
#        others - reserved
# 0x10 : Data signal of romcode
#        bit 31~0 - romcode[31:0] (Read/Write)
# 0x14 : Data signal of romcode
#        bit 31~0 - romcode[63:32] (Read/Write)
# 0x1c : Data signal of length_r
#        bit 31~0 - length_r[31:0] (Read/Write)

# Program physical address for the romcode base address
ipReadROMCODE.write(0x10, npROM.device_address)
ipReadROMCODE.write(0x14, 0)
# Program length of moving data
ipReadROMCODE.write(0x1C, rom_size_final)


# ipReadROMCODE start to move the data from rom_buffer to bram
ipReadROMCODE.write(0x00, 1) # IP Start
while (ipReadROMCODE.read(0x00) & 0x04) == 0x00: # wait for done
    continue

print("Write to bram done")

```

```
Write to bram done
```

jupyter caravel_fpga Last Checkpoint: 1 小時前 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

No kernel Trusted Python 3

Run Code Voilà

```
print("Write to bram done")

```

Write to bram done

In [6]:

```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0] (Read)
# 0x20 : Data signal of ps_mprj_out
#        bit 5~0 - ps_mprj_out[37:32] (Read)
#        others  - reserved
# 0x34 : Data signal of ps_mprj_en
#        bit 31~0 - ps_mprj_en[31:0] (Read)
# 0x38 : Data signal of ps_mprj_en
#        bit 5~0 - ps_mprj_en[37:32] (Read)
#        others  - reserved

print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))

```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0x8
0x20 =  0x0
0x34 =  0xfffffff7
0x38 =  0x3f
```

jupyter caravel_fpga Last Checkpoint: 1 小時前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [7]:
```
# Release Caravel reset
# 0x10 : Data signal of outpin_ctrl
#        bit 0  - outpin_ctrl[0] (Read/Write)
#        others - reserved
print (ipOUTPIN.read(0x10))
ipOUTPIN.write(0x10, 1)
print (ipOUTPIN.read(0x10))
```

```
0
1
```

In [8]:
```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0] (Read)
# 0x20 : Data signal of ps_mprj_out
#        bit 5~0 - ps_mprj_out[37:32] (Read)
#        others  - reserved
# 0x34 : Data signal of ps_mprj_en
#        bit 31~0 - ps_mprj_en[31:0] (Read)
# 0x38 : Data signal of ps_mprj_en
#        bit 5~0 - ps_mprj_en[37:32] (Read)
#        others  - reserved

print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab610008
0x20 =  0x2
0x34 =  0xfff7
0x38 =  0x37
```

- Execute “counter_la.hex”

In [4]:
```
# Create np with 8K/4 (4 bytes per index) size and be initiled to 0
rom_size_final = 0

# Allocate dram buffer will assign physical address to ip ipReadROMCODE
npROM = allocate(shape=(ROM_SIZE >> 2,), dtype=np.uint32)

# Initial it by 0
for index in range (ROM_SIZE >> 2):
    npROM[index] = 0

npROM_index = 0
npROM_offset = 0
#fiROM = open("counter_wb.hex", "r+")
fiROM = open("counter_la.hex", "r+")
#fiROM = open("gcd_la.hex", "r+")
```

由執行結果得知，在 0x1C 的位置上值為 0xab51。

In [8]:

```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0] (Read)
# 0x20 : Data signal of ps_mprj_out
#        bit 5~0 - ps_mprj_out[37:32] (Read)
#        others  - reserved
# 0x34 : Data signal of ps_mprj_en
#        bit 31~0 - ps_mprj_en[31:0] (Read)
# 0x38 : Data signal of ps_mprj_en
#        bit 5~0 - ps_mprj_en[37:32] (Read)
#        others  - reserved

print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab51f976
0x20 =  0x0
0x34 =  0x0
0x38 =  0x3f
```

- All on jupyter notebook

jupyter caravel_fpga Last Checkpoint: 3 分鐘前 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code Voilà

In [1]:

```
from __future__ import print_function

import sys
import numpy as np
from time import time
import matplotlib.pyplot as plt

sys.path.append('/home/xilinx')
from pynq import Overlay
from pynq import allocate

ROM_SIZE = 0x2000 #8K
```

In [2]:

```
ol = Overlay("/home/xilinx/jupyter_notebooks/ipy_fpga/caravel_fpga.bit")
#ol.ip_dict
```

In [3]:

```
ipOUTPIN = ol.output_pin_0
ipPS = ol.caravel_ps_0
ipReadROMCODE = ol.read_romcode_0
```

jupyter caravel_fpga Last Checkpoint: 4 分鐘前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [4]:

```python
# Create np with 8K/4 (4 bytes per index) size and be initiled to 0
rom_size_final = 0

# Allocate dram buffer will assign physical address to ip ipReadROMCODE
npROM = allocate(shape=(ROM_SIZE >> 2,), dtype=np.uint32)

# Initial it by 0
for index in range (ROM_SIZE >> 2):
    npROM[index] = 0

npROM_index = 0
npROM_offset = 0
#fiROM = open("counter_wb.hex", "r+")
fiROM = open("counter_la.hex", "r+")
#fiROM = open("gcd_la.hex", "r+")

for line in fiROM:
    # offset header
    if line.startswith('@'):
        # Ignore first char @
        npROM_offset = int(line[1:].strip(b'\x00'.decode()), base = 16)
        npROM_offset = npROM_offset >> 2 # 4byte per offset
        #print (npROM_offset)
        npROM_index = 0
        continue
    #print (line)

    # We suppose the data must be 32bit alignment
    buffer = 0
    bytecount = 0
    for line_byte in line.strip(b'\x00'.decode()).split():
        buffer += int(line_byte, base = 16) << (8 * bytecount)
        bytecount += 1
        # Collect 4 bytes, write to npROM
        if(bytecount == 4):
            npROM[npROM_offset + npROM_index] = buffer
            # Clear buffer and bytecount
            buffer = 0
            bytecount = 0
            npROM_index += 1
            #print (npROM_index)
            continue
    # Fill rest data if not alignment 4 bytes
    if (bytecount != 0):
        npROM[npROM_offset + npROM_index] = buffer
        npROM_index += 1
```

jupyter caravel_fpga Last Checkpoint: 4 分鐘前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```python
fiROM.close()

rom_size_final = npROM_offset + npROM_index
#print (rom_size_final)

#for data in npROM:
#    print (hex(data))
```

In [5]:

```python
# 0x00 : Control signals
#        bit 0  - ap_start (Read/Write/COH)
#        bit 1  - ap_done (Read/COR)
#        bit 2  - ap_idle (Read)
#        bit 3  - ap_ready (Read)
#        bit 7  - auto_restart (Read/Write)
#        others - reserved
# 0x10 : Data signal of romcode
#        bit 31~0 - romcode[31:0] (Read/Write)
# 0x14 : Data signal of romcode
#        bit 31~0 - romcode[63:32] (Read/Write)
# 0x1c : Data signal of length_r
#        bit 31~0 - length_r[31:0] (Read/Write)

# Program physical address for the romcode base address
ipReadROMCODE.write(0x10, npROM.device_address)
ipReadROMCODE.write(0x14, 0)
# Program length of moving data
ipReadROMCODE.write(0x1C, rom_size_final)


# ipReadROMCODE start to move the data from rom_buffer to bram
ipReadROMCODE.write(0x00, 1) # IP Start
while (ipReadROMCODE.read(0x00) & 0x04) == 0x00: # wait for done
    continue

print("Write to bram done")

```

```
Write to bram done
```

jupyter caravel_fpga Last Checkpoint: 5 分鐘前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
print("Write to bram done")

```

```
Write to bram done
```

In [6]:

```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0] (Read)
# 0x20 : Data signal of ps_mprj_out
#        bit 5~0 - ps_mprj_out[37:32] (Read)
#        others  - reserved
# 0x34 : Data signal of ps_mprj_en
#        bit 31~0 - ps_mprj_en[31:0] (Read)
# 0x38 : Data signal of ps_mprj_en
#        bit 5~0 - ps_mprj_en[37:32] (Read)
#        others  - reserved

print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))

```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0x8
0x20 =  0x0
0x34 =  0xffffffff7
0x38 =  0x3f
```

In [7]:

```
# Release Caravel reset
# 0x10 : Data signal of outpin_ctrl
#        bit 0  - outpin_ctrl[0] (Read/Write)
#        others - reserved
print (ipOUTPIN.read(0x10))
ipOUTPIN.write(0x10, 1)
print (ipOUTPIN.read(0x10))
```

```
0
1
```

jupyter caravel_fpga Last Checkpoint: 5 分鐘前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
0x38 =  0x3f
```

In [7]:

```
# Release Caravel reset
# 0x10 : Data signal of outpin_ctrl
#        bit 0  - outpin_ctrl[0] (Read/Write)
#        others - reserved
print (ipOUTPIN.read(0x10))
ipOUTPIN.write(0x10, 1)
print (ipOUTPIN.read(0x10))
```

```
0
1
```

In [8]:


```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0] (Read)
# 0x20 : Data signal of ps_mprj_out
#        bit 5~0 - ps_mprj_out[37:32] (Read)
#        others  - reserved
# 0x34 : Data signal of ps_mprj_en
#        bit 31~0 - ps_mprj_en[31:0] (Read)
# 0x38 : Data signal of ps_mprj_en
#        bit 5~0 - ps_mprj_en[37:32] (Read)
#        others  - reserved

print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```


```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab51d8e1
0x20 =  0x0
0x34 =  0x0
0x38 =  0x3f
```

➢ Execute “gcd_la.hex”

```
In [4]: # Create np with 8K/4 (4 bytes per index) size and be initiled to 0
        rom_size_final = 0

        # Allocate dram buffer will assign physical address to ip ipReadROMCODE
        npROM = allocate(shape=(ROM_SIZE >> 2,), dtype=np.uint32)

        # Initial it by 0
        for index in range (ROM_SIZE >> 2):
            npROM[index] = 0

        npROM_index = 0
        npROM_offset = 0
        #fiROM = open("counter_wb.hex", "r+")
        #fiROM = open("counter_la.hex", "r+")
        fiROM = open("gcd_la.hex", "r+")
```

由執行結果得知，在 0x1C 的位置上值為 0xab51。

```
In [8]: # Check MPRJ_IO input/out/en
        # 0x10 : Data signal of ps_mprj_in
        #        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
        # 0x14 : Data signal of ps_mprj_in
        #        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
        #        others  - reserved
        # 0x1c : Data signal of ps_mprj_out
        #        bit 31~0 - ps_mprj_out[31:0] (Read)
        # 0x20 : Data signal of ps_mprj_out
        #        bit 5~0 - ps_mprj_out[37:32] (Read)
        #        others  - reserved
        # 0x34 : Data signal of ps_mprj_en
        #        bit 31~0 - ps_mprj_en[31:0] (Read)
        # 0x38 : Data signal of ps_mprj_en
        #        bit 5~0 - ps_mprj_en[37:32] (Read)
        #        others  - reserved

        print ("0x10 = ", hex(ipPS.read(0x10)))
        print ("0x14 = ", hex(ipPS.read(0x14)))
        print ("0x1c = ", hex(ipPS.read(0x1c)))
        print ("0x20 = ", hex(ipPS.read(0x20)))
        print ("0x34 = ", hex(ipPS.read(0x34)))
        print ("0x38 = ", hex(ipPS.read(0x38)))

        0x10 =  0x0
        0x14 =  0x0
        0x1c =  0xab510041
        0x20 =  0x0
        0x34 =  0x0
        0x38 =  0x3f
```

■ All on jupyter notebook

jupyter caravel_fpga Last Checkpoint: 2023年11月2日 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code Voilà

In [1]:
```
from __future__ import print_function

import sys
import numpy as np
from time import time
import matplotlib.pyplot as plt

sys.path.append('/home/xilinx')
from pynq import Overlay
from pynq import allocate

ROM_SIZE = 0x2000 #8K
```

In [2]:
```
ol = Overlay("/home/xilinx/jupyter_notebooks/caravel_fpga.bit")
#ol.ip_dict
```

In [3]:
```
ipOUTPIN = ol.output_pin_0
ipPS = ol.caravel_ps_0
ipReadROMCODE = ol.read_romcode_0
```

jupyter caravel_fpga Last Checkpoint: 2023年11月2日 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code Voilà

In [4]:
```
# Create np with 8K/4 (4 bytes per index) size and be initiled to 0
rom_size_final = 0

# Allocate dram buffer will assign physical address to ip ipReadROMCODE
npROM = allocate(shape=(ROM_SIZE >> 2,), dtype=np.uint32)

# Initial it by 0
for index in range (ROM_SIZE >> 2):
    npROM[index] = 0

npROM_index = 0
npROM_offset = 0
#fiROM = open("counter_wb.hex", "r+")
#fiROM = open("counter_la.hex", "r+")
fiROM = open("gcd_la.hex", "r+")

for line in fiROM:
    # offset header
    if line.startswith('@'):
        # Ignore first char @
        npROM_offset = int(line[1:].strip(b'\x00'.decode()), base = 16)
        npROM_offset = npROM_offset >> 2 # 4byte per offset
        #print (npROM_offset)
        npROM_index = 0
        continue
    #print (line)

    # We suppose the data must be 32bit alignment
    buffer = 0
    bytecount = 0
    for line_byte in line.strip(b'\x00'.decode()).split():
        buffer += int(line_byte, base = 16) << (8 * bytecount)
        bytecount += 1
        # Collect 4 bytes, write to npROM
        if(bytecount == 4):
            npROM[npROM_offset + npROM_index] = buffer
            # Clear buffer and bytecount
            buffer = 0
            bytecount = 0
            npROM_index += 1
            #print (npROM_index)
            continue
```

jupyter caravel_fpga Last Checkpoint: 2023年11月2日 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
    # Fill rest data if not alignment 4 bytes
    if (bytecount != 0):
        npROM[npROM_offset + npROM_index] = buffer
        npROM_index += 1

fiROM.close()

rom_size_final = npROM_offset + npROM_index
#print (rom_size_final)

#for data in npROM:
#    print (hex(data))
```

In [5]:

```
# 0x00 : Control signals
#        bit 0  - ap_start (Read/Write/COH)
#        bit 1  - ap_done (Read/COR)
#        bit 2  - ap_idle (Read)
#        bit 3  - ap_ready (Read)
#        bit 7  - auto_restart (Read/Write)
#        others - reserved
# 0x10 : Data signal of romcode
#        bit 31~0 - romcode[31:0] (Read/Write)
# 0x14 : Data signal of romcode
#        bit 31~0 - romcode[63:32] (Read/Write)
# 0x1c : Data signal of length_r
#        bit 31~0 - length_r[31:0] (Read/Write)

# Program physical address for the romcode base address
ipReadROMCODE.write(0x10, npROM.device_address)
ipReadROMCODE.write(0x14, 0)
# Program length of moving data
ipReadROMCODE.write(0x1C, rom_size_final)


# ipReadROMCODE start to move the data from rom_buffer to bram
ipReadROMCODE.write(0x00, 1) # IP Start
while (ipReadROMCODE.read(0x00) & 0x04) == 0x00: # wait for done
    continue

print("Write to bram done")
```

```
Write to bram done
```

jupyter caravel_fpga Last Checkpoint: 2023年11月2日 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [6]:

```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0] (Read)
# 0x20 : Data signal of ps_mprj_out
#        bit 5~0 - ps_mprj_out[37:32] (Read)
#        others  - reserved
# 0x34 : Data signal of ps_mprj_en
#        bit 31~0 - ps_mprj_en[31:0] (Read)
# 0x38 : Data signal of ps_mprj_en
#        bit 5~0 - ps_mprj_en[37:32] (Read)
#        others  - reserved

print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0x8
0x20 =  0x0
0x34 =  0xffffff7
0x38 =  0x3f
```

In [7]:

```
# Release Caravel reset
# 0x10 : Data signal of outpin_ctrl
#        bit 0  - outpin_ctrl[0] (Read/Write)
#        others - reserved
print (ipOUTPIN.read(0x10))
ipOUTPIN.write(0x10, 1)
print (ipOUTPIN.read(0x10))
```

```
0
1
```

jupyter caravel_fpga Last Checkpoint: 2023年11月2日 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code Voilà

In [8]:

```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0] (Read)
# 0x20 : Data signal of ps_mprj_out
#        bit 5~0 - ps_mprj_out[37:32] (Read)
#        others  - reserved
# 0x34 : Data signal of ps_mprj_en
#        bit 31~0 - ps_mprj_en[31:0] (Read)
# 0x38 : Data signal of ps_mprj_en
#        bit 5~0 - ps_mprj_en[37:32] (Read)
#        others  - reserved

print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab510041
0x20 =  0x0
0x34 =  0x0
0x38 =  0x3f
```

Study caravel_fpga.ipynb, and be familiar with caravel SoC control flow:

此次 Lab5 是由租借 FPGA 板執行 pythin code 來進行驗證，透過 ip 來完成資料的傳遞。

透過 read_romcode ip 來將 program.hex 傳遞。將資料從 DDR 內傳遞至硬體 BRAM 內。

透過 RsetControl ip 來溝通 PS 與 Caravel 內的 MPRJ_IO。用此 ip 設定 reset 信號控制，1 或 0 分別控制 Caravel 的 reset pin 是 assert 還是 de-assert。

透過 MPRJ_IO 經由 caravel_ps 這個 ip 在執行完後溝通。提供 AXI Lite 接口供 PS CPU 讀取 MPRJ_IO/OUT/EN bit，用 HLS 實現並導出 IP 以供 Vivado 使用。