國立台灣大學電機資訊學院電子工程學研究所

系統晶片設計實驗
Soc Design Laboratory

Lab5 Report

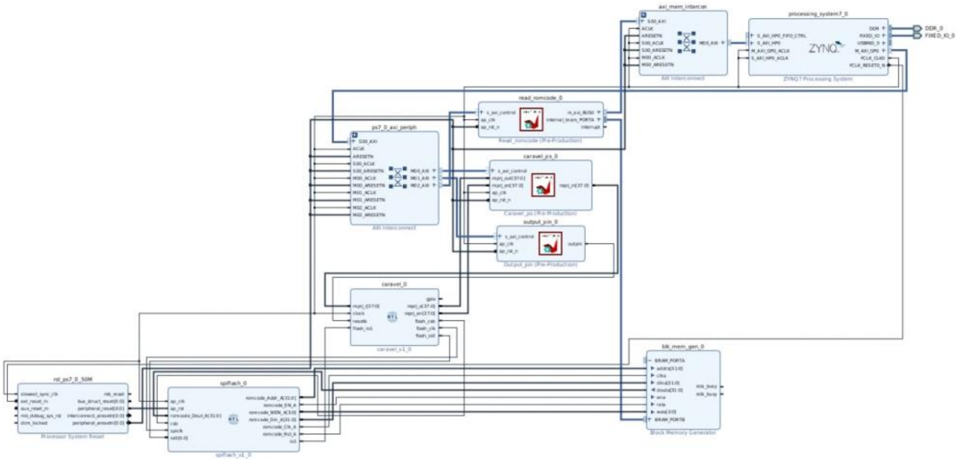
Caravel SOC - Caravel FPGA

學生： M11202109 蘇柏丞
M11202103 陳泓宇
M11202207 呂彥霖

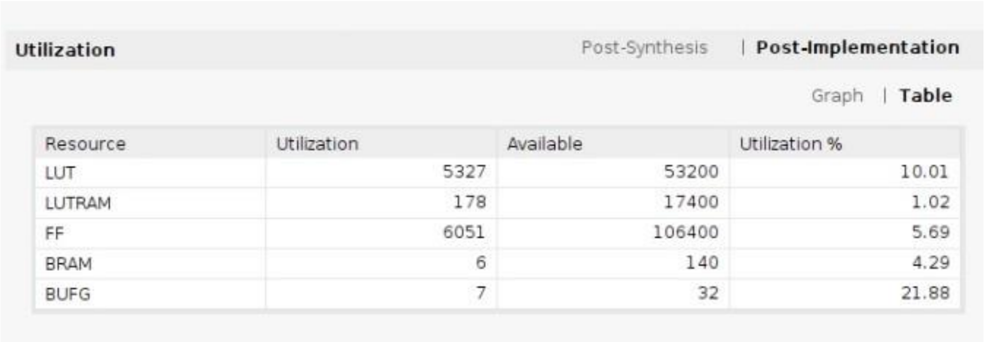
老師： 賴 瑾

中華民國 112 年 11 月 22 日

一、 Block diagram：



二、 FPGA utilization：



Utilization | Post-Synthesis | **Post-Implementation**

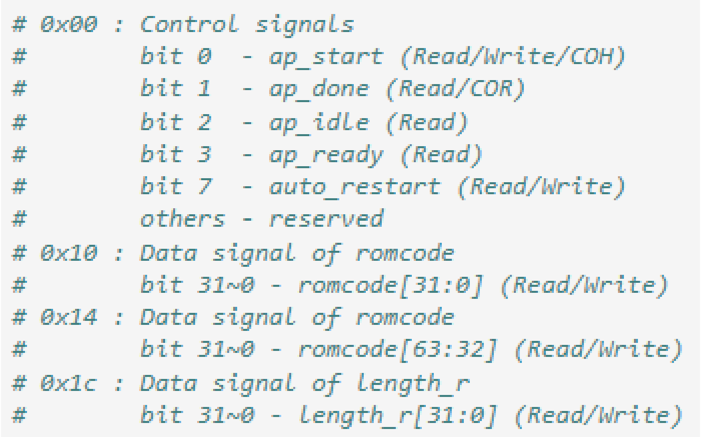
Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	5327	53200	10.01
LUTRAM	178	17400	1.02
FF	6051	106400	5.69
BRAM	6	140	4.29
BUFG	7	32	21.88

三、 Function of IP：

1. Read_romcode

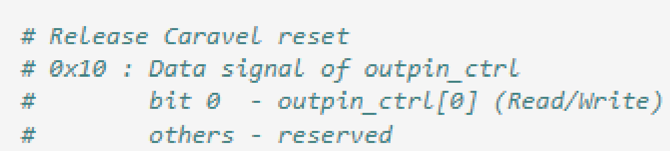
用來將 fireware code 從 DRAM 轉移至 BRAM，因為我們的 BRAM 有大小限制，故在設計時 fireware code 限制在 8K 以下，並實現 axilite 的 address map，當 0x00 時是用於控制 ap 訊號，當 0x10 和 0x14 時是控制 BRAM buffer 的 base address 訊號，當 0x1c 時控制 data length 訊號。



```
# 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)
```

2. ResetControl

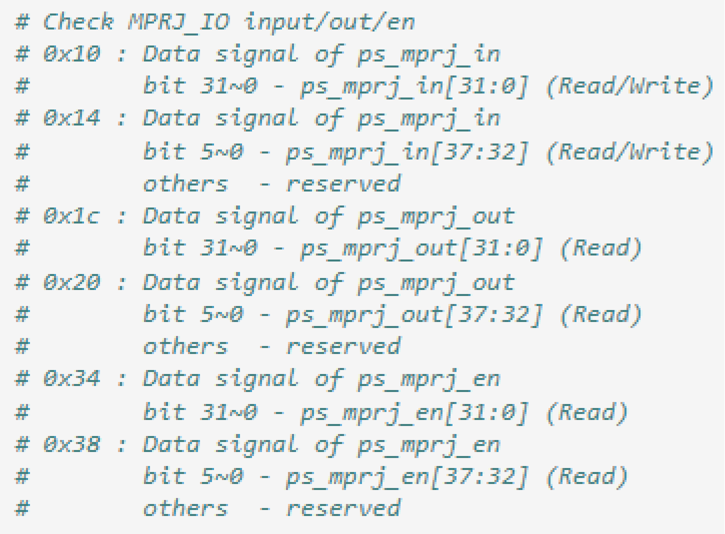
用於控制 Caravel 的 reset pin，並實現 axilite 的 address map 控制 PS CPU 的輸出，當 0x10 時根據 bit 0 控制 outpin_ctrl，進而控制 caravel 的 reset pin。



```
# Release Caravel reset
# 0x10 : Data signal of outpin_ctrl
#         bit 0   - outpin_ctrl[0] (Read/Write)
#         others - reserved
```

3. Caravel_ps

用來實現 GPIO 的 Monitor，本實驗當中用 jupyter code 實現 MMIO 去 read Register，進而讀取 Caravel 產生出來的 Signal data 是否正確。



```
# 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
```

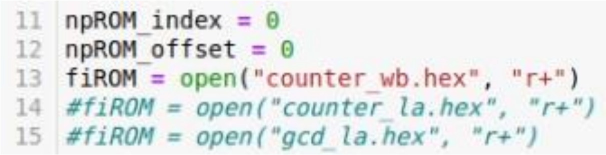
4. Spiflash

用來實現 SPI slave device，在本實驗中只用於讀取 Caravel 傳送過來 read 指令(0x30)，讀到指令後回傳 fireware code 回去，當中的 fireware code 是從 BRAM 存取。

四、 Workload

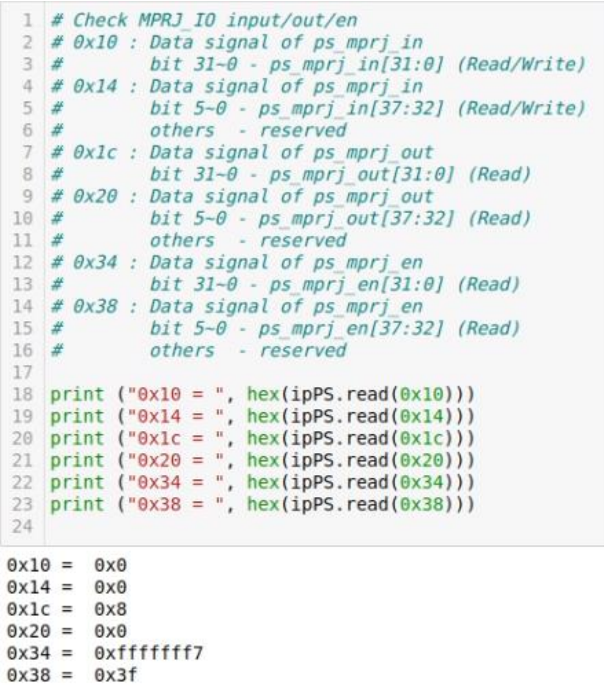
1. counter_wb.hex

- read workload



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

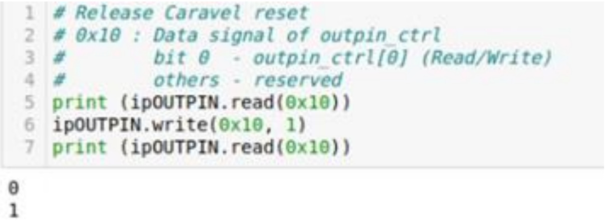
- Check MPRJ_IO after bram has been written



```
# 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)))

```

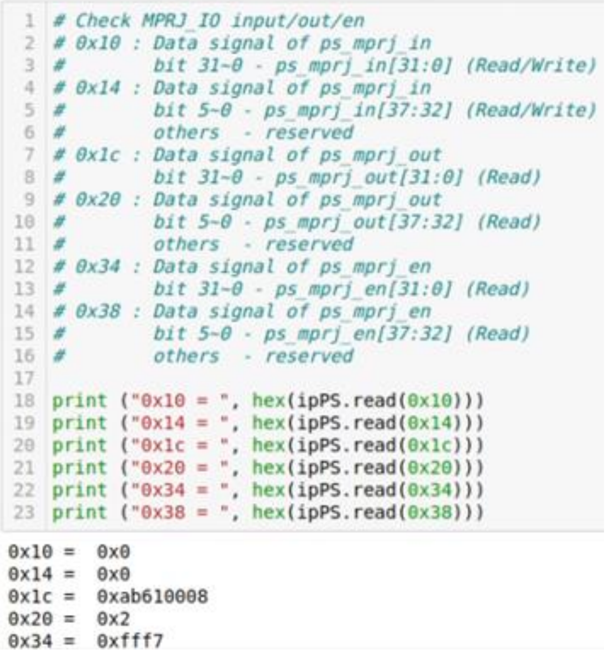
```
0x10 =  0x0
0x14 =  0x0
0x1c =  0x8
0x20 =  0x0
0x34 =  0xfffffff7
0x38 =  0x3f
```

- de-assert Caravel reset pin and Get mprj_i/o/en data



```
# 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))
```

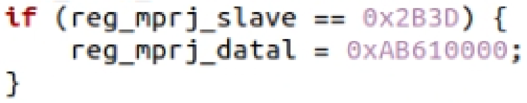
```
0
1
```

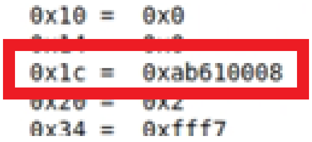


```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#        bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#        bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#        others  - reserved
# 0x1c : Data signal of ps_mprj_out
#        bit 31~0 - ps_mprj_out[31:0]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

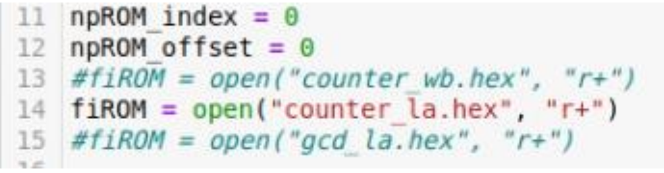
- Compare the mprj_o value with final result in the firmware code

firmware code	
	`if (reg_mprj_slave == 0x2B3D) {`
	`    reg_mprj_datal = 0xAB610000;`
	`}`

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

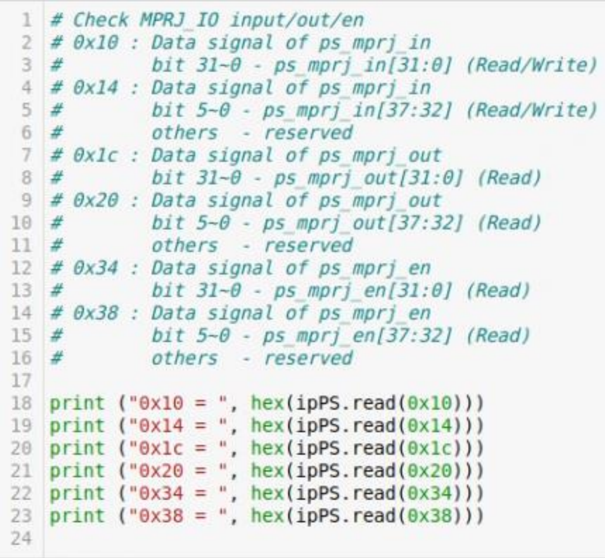
2. counter_la.hex

- read workload

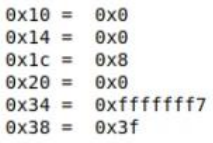


```
npROM_index = 0
npROM_offset = 0
#fiROM = open("counter_wb.hex", "r+")
fiROM = open("counter_la.hex", "r+")
#fiROM = open("gcd_la.hex", "r+")
```

- Check MPRJ_IO after bram has been written

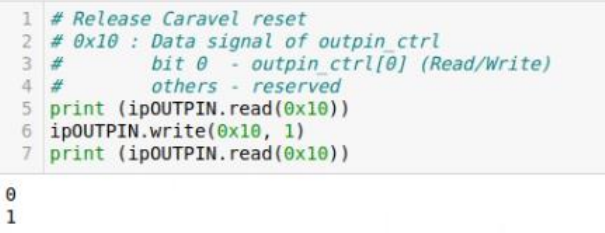


```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#         bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#         bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#         others  - reserved
# 0x1c : Data signal of ps_mprj_out
#         bit 31~0 - ps_mprj_out[31:0]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

```



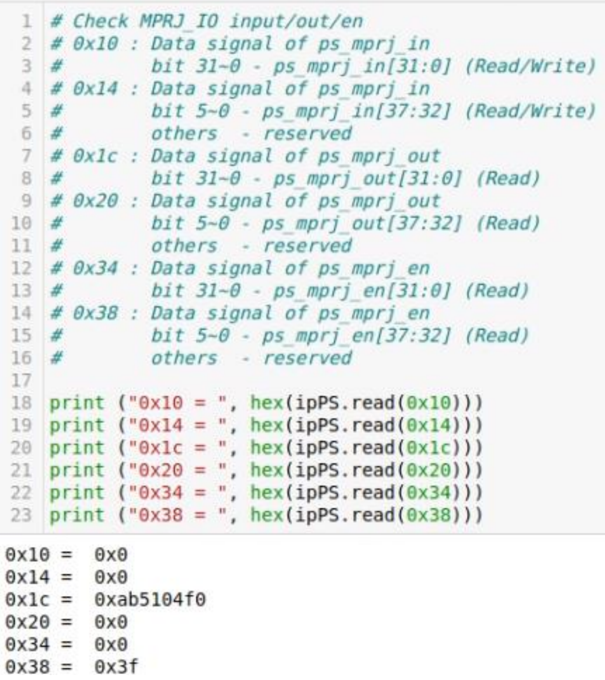
```
0x10 =  0x0
0x14 =  0x0
0x1c =  0x8
0x20 =  0x0
0x34 =  0xfffffff7
0x38 =  0x3f
```

- de-assert Caravel reset pin and Get mprj_i/o/en data



```
# 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))
```

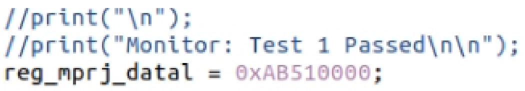
```
0
1
```

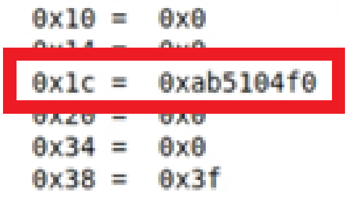


```
# Check MPRJ_IO input/out/en
# 0x10 : Data signal of ps_mprj_in
#         bit 31~0 - ps_mprj_in[31:0] (Read/Write)
# 0x14 : Data signal of ps_mprj_in
#         bit 5~0 - ps_mprj_in[37:32] (Read/Write)
#         others  - reserved
# 0x1c : Data signal of ps_mprj_out
#         bit 31~0 - ps_mprj_out[31:0] (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 =  0xab5104f0
0x20 =  0x0
0x34 =  0x0
0x38 =  0x3f
```

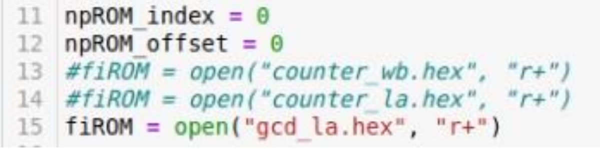
- Compare the mprj_o value with final result in the firmware code

Firware code	
	//print("\n");
	//print("Monitor: Test 1 Passed\n\n");
	reg_mprj_datal = 0xAB510000;

mprj_o value	 0x10 = 0x0 0x14 = 0x0 0x1c = 0xab5104f0 0x20 = 0x0 0x34 = 0x0 0x38 = 0x3f

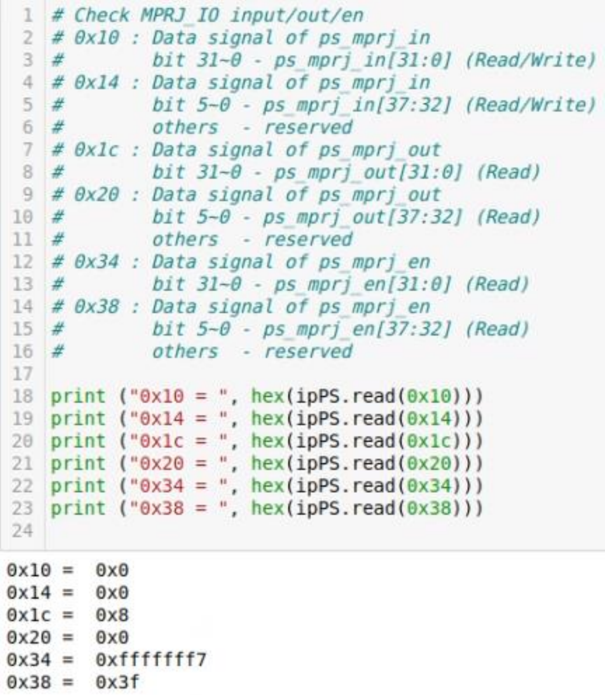
3. gcd_la.hex

- read workload



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

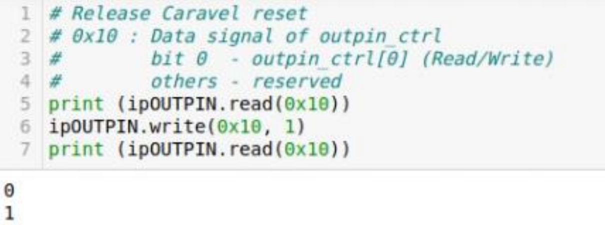
- Check MPRJ_IO after bram has been written



```
# 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)))

```

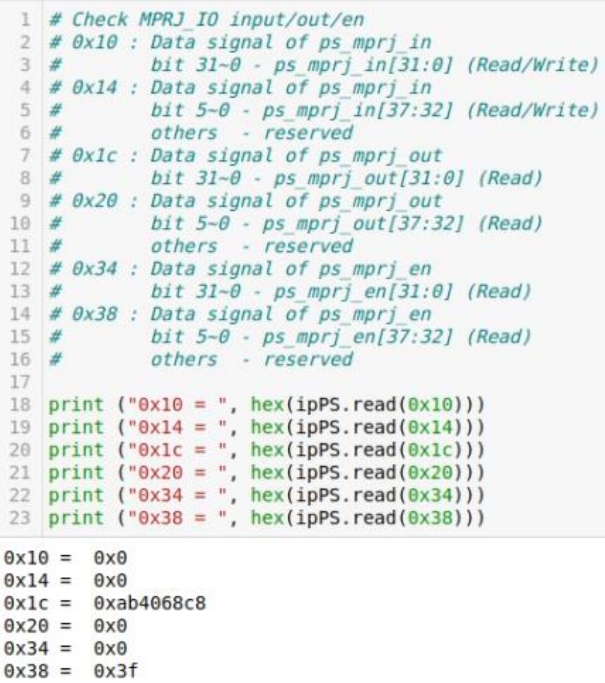
```
0x10 =  0x0
0x14 =  0x0
0x1c =  0x8
0x20 =  0x0
0x34 =  0xfffffff7
0x38 =  0x3f
```

- de-assert Caravel reset pin and Get mprj_i/o/en data



```
# 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))
```

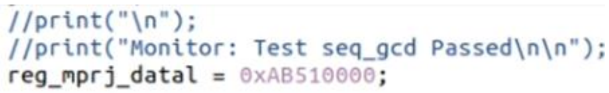
```
0
1
```



```
# 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 =  0xab4068c8
0x20 =  0x0
0x34 =  0x0
0x38 =  0x3f
```

- mprj_o value



```
//print("\n");
//print("Monitor: Test seq_gcd Passed\n\n");
reg_mprj_datal = 0xAB510000;
```