## Overall system:

此 lab 的 kernel function 很簡單,即是輸入2個 input,然後返回相乘的結果。不過要利用 AXILite adaptor 中的 Register 當作 input 傳遞的工具,傳入 IN1、IN2 進去,得到的結果也用 AXILite adaptor 的 Register 存起來。然後 Register 中,input 的傳入是利用 AXILite 的 AW 和 W,將資料寫入 Register。而讀取返回的結果則是利用 AXILite 的 AR 和 R,將 output 從 Register 讀出來。

## What is observed & learned:

學到 AXILite 協議的內容,知道 master 和 slave 是怎麼溝通的。也知道怎麼將東西包成 IP,相比於暑假自學時,也更了解這次實驗背後的原理。

## Screen dump:

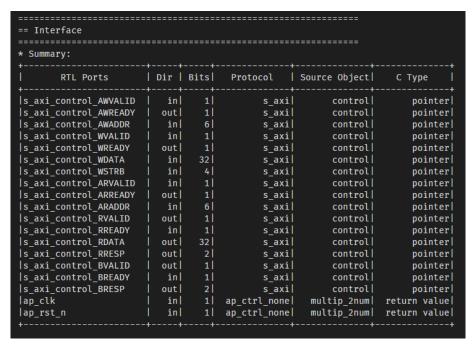
## 1. Performance

```
== Performance Estimates
Timing:
  * Summary:
  | Clock | Target | Estimated | Uncertainty |
  |ap_clk | 10.00 ns| 6.912 ns|
                                      2.70 ns
  * Summary:
     Latency (cycles) |
                        Latency (absolute)
                                             | Interval | Pipeline|
                                             | min | max |
                                                           Type
                    3| 30.000 ns| 30.000 ns|
  + Detail:
      * Instance:
      * Loop:
```

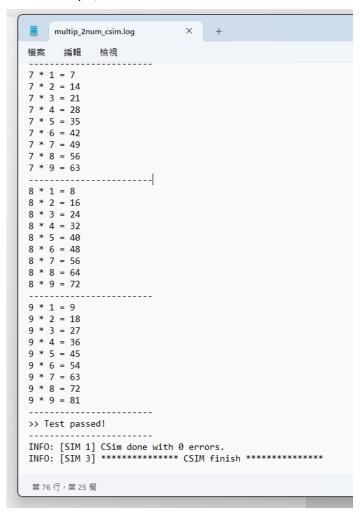
#### 2. Utilization

======================================	======= imates ========	======	:======	:======	:=====
* Summary:					
Name	   BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-i
Expression	l -I	-	-	-	-1
FIFO	-	-	-	-	-I
Instance	0	3	309	282	- I
Memory	-	-	-	-	-
Multiplexer	-	-	-	25	-
Register	-	-	100	-1	-1
+  Total	0	3	409	307	0
Available	280	220	106400	53200	øİ
Utilization (%)	0	1	~0	~0	øl
+	+				+

## 3. Interface



## 4. Co-simulation transcript /waveform



# 5. Jupyter Notebook execution results