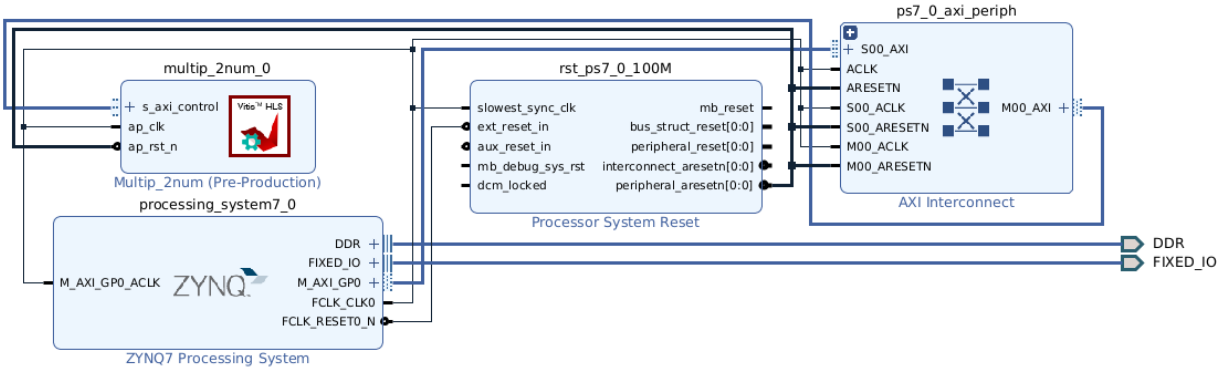
Lab1 Report

1. Introduction about the overall system



Lab1 為基礎環境的建置與乘法在硬體(PYNQ Z2)上的實作。透過 Vitis_hls 撰寫 pragma 來定義輸入與輸出透過何種 program I/O 來傳輸，並使用先前寫好的 testbench 對於主程式進行 C simulation。在 Simulation 後，進行 Synthesis，觀察在主視窗跳出的 synthesis report 並分析其 timing 和 area 的 information。接著進行 C 和 RTL 的 Cosimulation 觀察波型，並輸出 RTL code 供 Vivado 進行下一步動作。

由 Vitis_hls 所匯出的 RTL code 匯入 Vivado，並對其進行 block design 和加入 ZYNQ7 Processing System 作為 host 端。進行完 block design 後，對其進行 wrapper 以利後續 Bitstream 的生成。最後將 Bitstream 輸出的.bit 檔、.hwh 檔和 host program 上傳至 OnlineFPGA 的 Jupyter Notebook 中，在 PYNQ 上進行運算，並與先前 Vitis_hls 的 C Simulation 結果進行比較。

2. Observation & Learning

(1) 建製實驗環境(ubuntu)遇到的問題與學習到的指令

在建置 ubuntu 的環境時，觀察出每一個 command 都有它的含意，如 mkdir 為建立資料夾所使用的指令、ls 為列出工作路徑下的檔案或是目錄清單的指令、cd 為切換到目錄的路徑的指令、cp 為拷貝檔案或目錄的指令和 sudo 指令為 superuser do，用以暫時提升管理權限進行一些需要 root 權限的指令。

依照安裝說明書操作時，我發現到一些指令無效指令，上網搜尋後得到一些見解，在以下列出。在分割硬碟時，需使用 fdisk 指令來處理，然而 fdisk 需要 root 權限才能進行，故若要切割 sdb 硬碟，需使用 sudo fdisk /dev/sdb 指令。在安裝 Vitis 前的設置部分，欲了解所有可更新 software 需使用 sudo apt update 指令。在安裝 Vitis 時，因著 /tools/Xilinx 的還未建立，系統將會提示是否自動建立資料夾，然而使用自動建立資料夾造成在安裝完成後，執行 echo 'source /tools/Xilinx/Vitis/2022.1/settings64.sh' sudo tee -a ~/.bashrc 指令時將找不到 source /tools/Xilinx/Vitis/2022.1/settings64.sh 指令的路徑，故後來我使用 mkdir 建立資料夾來解決此問題。

(2) Vitis_hls

HLS 為 High Level Synthesis 的縮寫，其為透過高階程式語言(如 C 語言)的形式來表述硬體的行為，並透過 pragma 來定義每個 interface 之間所使用的 protocol(通訊協定)。在 Vitis_hls 中，所有的 interface 都是使用 axi protocol 來規定的，且 protocol 可以大致區分為 block level protocol 和 port level protocol。

```
1
2  #include "Multiplication.h"
3
4  void multip_2num(int32_t n32In1, int32_t n32In2, int32_t* pn32ResOut)
5  {
6  #pragma HLS INTERFACE s_axilite port=pn32ResOut
7  #pragma HLS INTERFACE s_axilite port=n32In2
8  #pragma HLS INTERFACE s_axilite port=n32In1
9  #pragma HLS INTERFACE ap_ctrl_none port=return
10
11     *pn32ResOut = n32In1 * n32In2;
12
13     return;
14 }
```

本次實驗中，block level protocol 使用的為 data-driven 的 ap_ctrl_none。此實驗中共有兩個 input port 和一個 output port，都是使用 s_axilite 作為 port level protocol。在 testbench 部分則是匯入標頭檔後，對於乘數位於 1~9 和被乘數位於 1~9 的乘法進行測試，若結果正確輸出”Test passed!”，反之則輸出”Test failed!”。

在了解程式碼的運作方式後對於其進行 C Simulation，可透過 testbench 對於 C code 進行 behavior 的驗證，如下圖 testbench 片段所示。

```
76 ------------------------
77 8 * 1 = 8
78 8 * 2 = 16
79 8 * 3 = 24
80 8 * 4 = 32
81 8 * 5 = 40
82 8 * 6 = 48
83 8 * 7 = 56
84 8 * 8 = 64
85 8 * 9 = 72
86 ------------------------
87 9 * 1 = 9
88 9 * 2 = 18
89 9 * 3 = 27
90 9 * 4 = 36
91 9 * 5 = 45
92 9 * 6 = 54
93 9 * 7 = 63
94 9 * 8 = 72
95 9 * 9 = 81
96 ------------------------
97 >> Test passed!
98 ------------------------
99 INFO: [SIM 1] CSim done with 0 errors.
100 INFO: [SIM 3] *************** CSIM finish ***************
```

在進行完成 C simulation 後，將對於 C code 進行 Synthesis。

Latency(cycles)	Latency(ns)
3	30.000

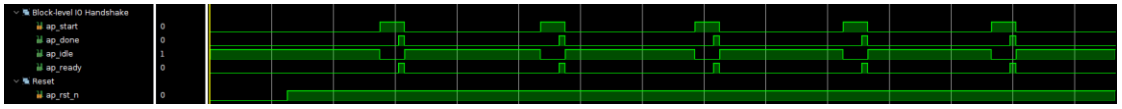
在 Performance 部分，在系統頻率 100MHz 的環境下共花了 3 個 cycles / 30 ns 來完成此次的任務。而在 Utilization 部分，共使用了 3 個 DSP(Digital System Processing)、409 個 FF(Flip-Flop)和 307 個 LUT(Look Up Table)來建構此運算需要的 resource，如下圖所示。

DSP	FF	LUT
3	409	307

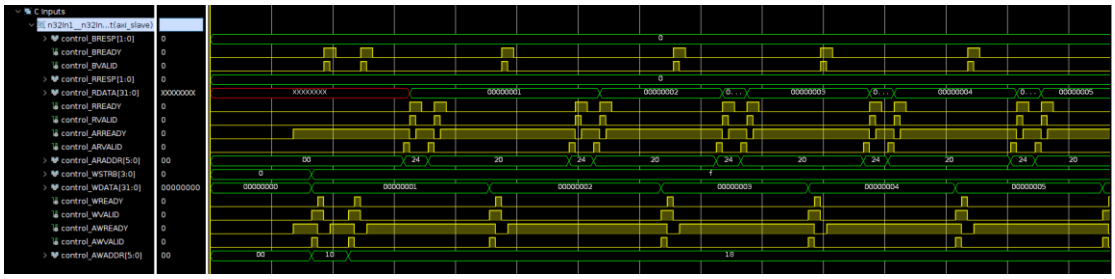
在 C synthesis 的部分，也可以觀察每個 I/O port 系統給予對應到的 configuration register，如 Input Port n32ln1 被分配到 0x10 的 register、Input Port n32ln2 被分配到 0x18 的 register、Output Port pn32ResOut 被分配到 0x20 的 register 和系統指定的 pn32ResOut_Ctrl 被分配到 0x24 的 register，如下圖所示。

Interface	Register	Offset	Width	Access	Description	Bit Fields
s_axi_control	n32In1	0x10	32	W	Data signal of n32In1	
s_axi_control	n32In2	0x18	32	W	Data signal of n32In2	
s_axi_control	pn32ResOut	0x20	32	R	Data signal of pn32ResOut	
s_axi_control	pn32ResOut_ctrl	0x24	32	R	Control signal of pn32ResOut	0=pn32ResOut_ap_vld

在完成 C Synthesis 後，進行 C 和 RTL 的 Co-simulation，可得整個系統的 Waveform，如下圖所示。



上圖為 block level protocol 的 Waveform，綠色波型依序為 ap_start、ap_done、ap_idle、ap_ready 和 ap_rst_n，在要開始執行時 ap_rst_n 由 0 升至 1 等待開始運作。隨著 ap_start 的抬升，ap_idle 離開 ap_idle 狀態由 1 轉乘 0。當 ap_ready 由 0 抬升至 1，ap_done 也隨之升起，此時 Master 端與 Slave 端開始 handshake。而當 ap_ready 下降至 0 時，ap_start 與 ap_done 隨之下降至 0，ap_idle 則抬升至 1 等待下一次 handshake。



下半部分為 port level protocol 的 Waveform，可觀察到在 AW(address write)和 AR(address read)的部分，ready 的訊號都打得比 valid 的訊號早，我猜測前面可能有接 buffer 才能達到此種效果。觀察 R(read)和 W(write)訊號可發現兩個動作時序排序方式不太一樣，read 的部分為在取得要讀取的資料位址後才發送 Rvalid 訊號，而 write 的則為先發出 Wvalid，當取得寫入資料位址時，直接將 Wready 訊號拉為 1 進行寫入。

在完成 C 和 RTL 的 Co-simulation 後，匯出 RTL code 供 Vivado 進一步分析使用。

(3) Vivado

由 Vitis_hls 匯出的 RTL code，在 Vivado 中可建立其 block design，並將設計好的 kernel 與 host 端(ZYNQ7 Processing System)接線，而後對其 wrapper 並生成 BitStream。

(4) Jupyter Notebook

由 Vivado 輸出的 BitStream 中的.bit 檔、.hwh 檔與寫好的.ipynb 檔，將其上傳至 OnlineFPGA 的 Jupyter Notebook 中並在 PYZQ Z2 上進

行運算得到結果，並與 C Simulation 進行比較。

3. Screen dump

(1) Performance

```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+----------+----------+------------+
    |  Clock |  Target  | Estimated| Uncertainty|
    +--------+----------+----------+------------+
    |ap_clk  |  10.00 ns|  6.912 ns|     2.70 ns|
    +--------+----------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+-----------+-----------+-----+-----+---------+
    |  Latency (cycles) |   Latency (absolute)  |  Interval | Pipeline|
    |   min   |   max   |    min    |    max    | min | max |   Type  |
    +---------+---------+-----------+-----------+-----+-----+---------+
    |        3|        3|  30.000 ns|  30.000 ns|    4|    4|       no|
    +---------+---------+-----------+-----------+-----+-----+---------+

    + Detail:
        * Instance:
        N/A

        * Loop:
        N/A
```

(2) Utilization

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

+ Detail:
    * Instance:
    +----------------------+--------------------+---------+----+-----+-----+-----+
    |        Instance      |       Module       | BRAM_18K| DSP|  FF | LUT | URAM|
    +----------------------+--------------------+---------+----+-----+-----+-----+
    |control_s_axi_U       |control_s_axi       |        0|   0|  144|  232|    0|
    |mul_32s_32s_32_2_1_U1 |mul_32s_32s_32_2_1  |        0|   3|  165|   50|    0|
    +----------------------+--------------------+---------+----+-----+-----+-----+
    |Total                 |                    |        0|   3|  309|  282|    0|
    +----------------------+--------------------+---------+----+-----+-----+-----+
```

```
    * DSP:
    N/A

    * Memory:
    N/A

    * FIFO:
    N/A

    * Expression:
    N/A

    * Multiplexer:
    +-----------+----+-----------+-----+-----------+
    |    Name   | LUT| Input Size| Bits| Total Bits|
    +-----------+----+-----------+-----+-----------+
    |ap_NS_fsm  |  25|          5|    1|          5|
    +-----------+----+-----------+-----+-----------+
    |Total      |  25|          5|    1|          5|
    +-----------+----+-----------+-----+-----------+

    * Register:
    +--------------------+----+----+-----+-----------+
    |        Name        | FF | LUT| Bits| Const Bits|
    +--------------------+----+----+-----+-----------+
    |ap_CS_fsm           |   4|   0|    4|          0|
    |mul_ln11_reg_69     |  32|   0|   32|          0|
    |n32In1_read_reg_64  |  32|   0|   32|          0|
    |n32In2_read_reg_59  |  32|   0|   32|          0|
    +--------------------+----+----+-----+-----------+
    |Total               | 100|   0|  100|          0|
    +--------------------+----+----+-----+-----------+
```

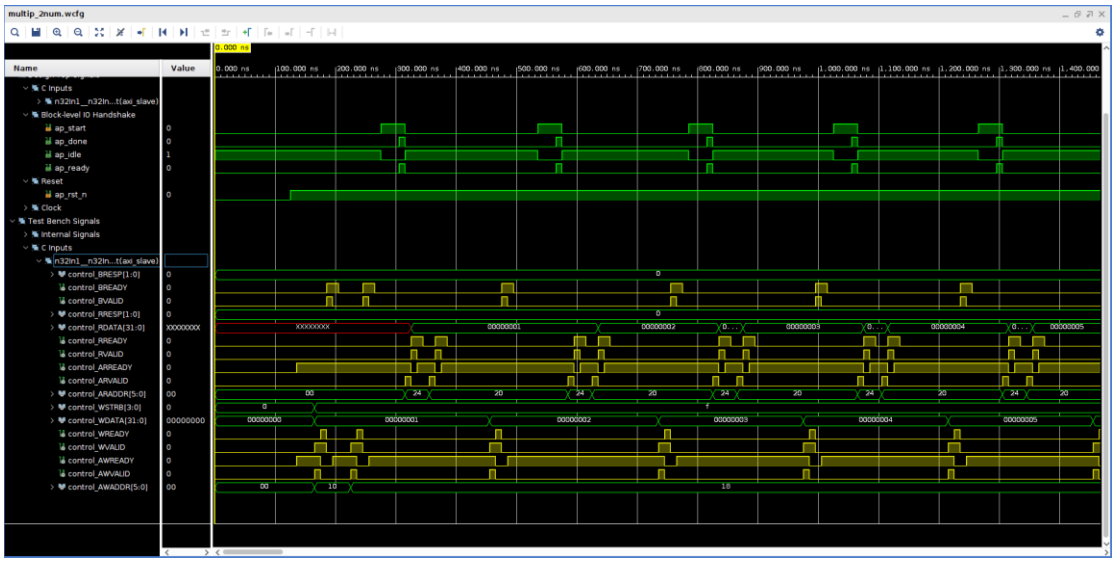
(3) Interface

```
== Interface
* Summary:
+-----------------------+-----+-----+------------+--------------+--------------+
|       RTL Ports       | Dir | Bits|  Protocol  | Source Object|    C Type    |
+-----------------------+-----+-----+------------+--------------+--------------+
|s_axi_control_AWVALID  |   in|    1|       s_axi|       control|       pointer|
|s_axi_control_AWREADY  |  out|    1|       s_axi|       control|       pointer|
|s_axi_control_AWADDR   |   in|    6|       s_axi|       control|       pointer|
|s_axi_control_WVALID   |   in|    1|       s_axi|       control|       pointer|
|s_axi_control_WREADY   |  out|    1|       s_axi|       control|       pointer|
|s_axi_control_WDATA    |   in|   32|       s_axi|       control|       pointer|
|s_axi_control_WSTRB    |   in|    4|       s_axi|       control|       pointer|
|s_axi_control_ARVALID  |   in|    1|       s_axi|       control|       pointer|
|s_axi_control_ARREADY  |  out|    1|       s_axi|       control|       pointer|
|s_axi_control_ARADDR   |   in|    6|       s_axi|       control|       pointer|
|s_axi_control_RVALID   |  out|    1|       s_axi|       control|       pointer|
|s_axi_control_RREADY   |   in|    1|       s_axi|       control|       pointer|
|s_axi_control_RDATA    |  out|   32|       s_axi|       control|       pointer|
|s_axi_control_RRESP    |  out|    2|       s_axi|       control|       pointer|
|s_axi_control_BVALID   |  out|    1|       s_axi|       control|       pointer|
|s_axi_control_BREADY   |   in|    1|       s_axi|       control|       pointer|
|s_axi_control_BRESP    |  out|    2|       s_axi|       control|       pointer|
|ap_clk                 |   in|    1|  ap_ctrl_hs|   multip_2num|  return value|
|ap_rst_n               |   in|    1|  ap_ctrl_hs|   multip_2num|  return value|
|ap_start               |   in|    1|  ap_ctrl_hs|   multip_2num|  return value|
|ap_done                |  out|    1|  ap_ctrl_hs|   multip_2num|  return value|
|ap_idle                |  out|    1|  ap_ctrl_hs|   multip_2num|  return value|
```

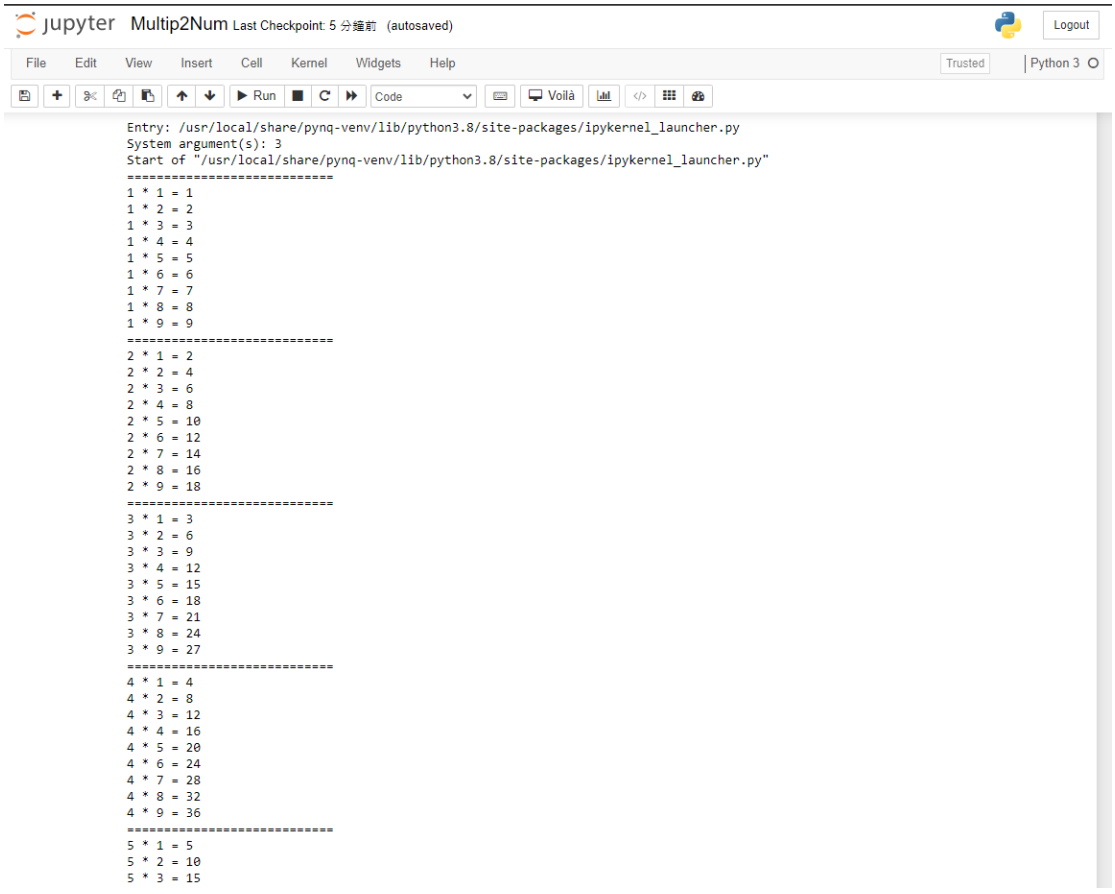
(4) Co-simulation transcript/waveform

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../hls_Mutiplication/Multiplication.cpp in debug mode
   Generating csim.exe
>> Start test!
------------------------
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
------------------------
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
------------------------
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
------------------------
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
------------------------
```

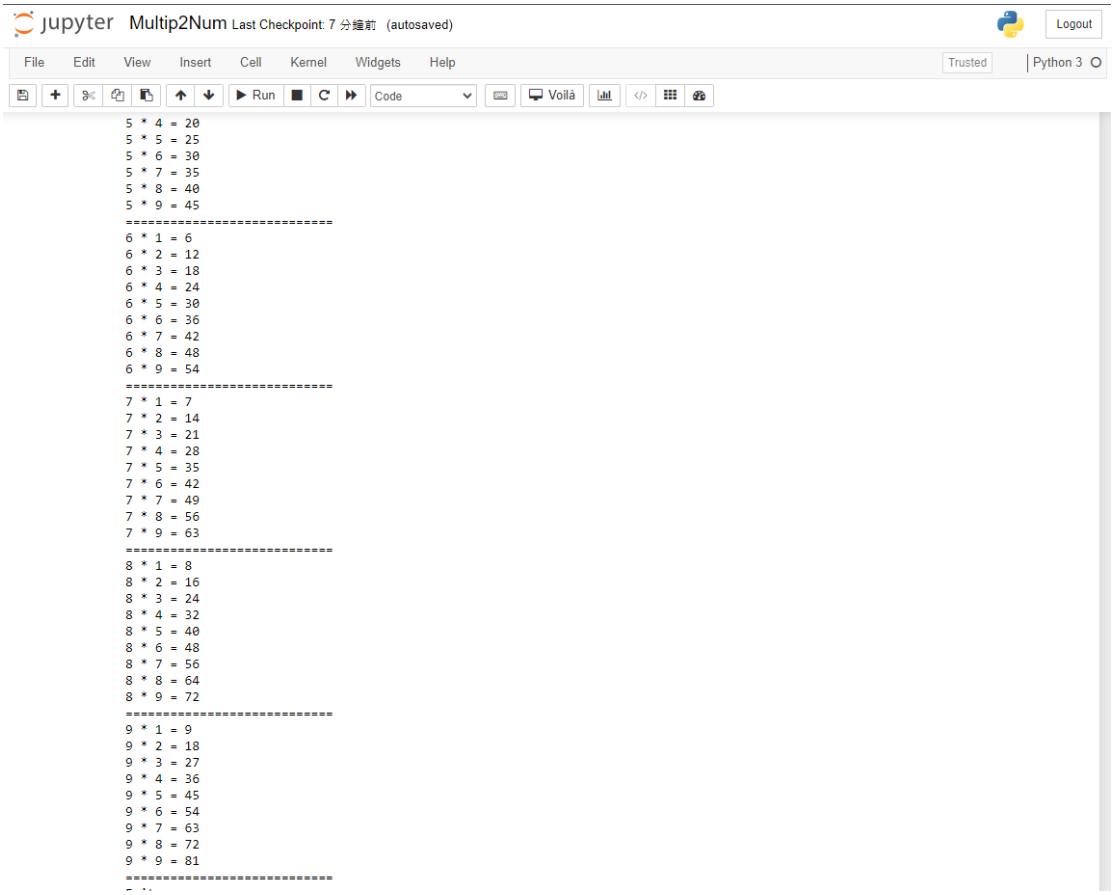
```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
-------------------------
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
-------------------------
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
-------------------------
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
-------------------------

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
-------------------------
>> Test passed!
```



(5) Jupyter Notebook execution results




```
Entry: /usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py
System argument(s): 3
Start of "/usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py"
============================
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
============================
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
============================
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
============================
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
============================
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
```


jupyter Multip2Num Last Checkpoint: 7 分鐘前 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Voilà
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
==========
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
==========
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
==========
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
==========
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
==========