

[SoC Lab] Lab1

tags: SoC Lab, SOC Design

Student ID Name

311551095 林聖博

附上此篇Hackmd Link : <https://hackmd.io/@Sheng08/H1hkiwMI6>

- [SoC Lab] Lab1
 - Lab 1
 - Tools Install Result
 - Introduction about the overall system
 - Step
 - 系統概述
 - Screen dump
 - Screen dump (Other)
 - Note
 - 名詞解釋
 - 觀念補充
 - 心得
 - 未來預期目標
 - Troubleshooting
 - 補充

Lab 1

- https://github.com/bol-edu/course-lab_1

Tools Install Result

- vitis

```
ubuntu@ubuntu2004:~$ vitis
***** Xilinx Vitis Development Environment
***** Vitis v2022.1 (64-bit)
**** SW Build 3524922 on 2022-04-14-18:00:18
** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

Launching Vitis with command /tools/Xilinx/Vitis/2022.1/eclipse/lnx64.o/eclipse -vmargs -Xms64m -Xmx1024m -Dorg.eclipse.swt.internal.gtk.cairoGraphics=false -Dosgi.configuration.area=@user.home/.Xilinx/Vitis/2022.1 --add-modules=ALL-SYSTEM --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.desktop/sun.swing=ALL-UNNAMED --add-opens=java.desktop/javax.swing=ALL-UNNAMED --add-opens=java.desktop/javafx.swing.tree=ALL-UNNAMED --add-opens=java.desktop/javafx.swing.plaf.basic=ALL-UNNAMED --add-opens=java.desktop/javafx.swing.plaf.synth=ALL-UNNAMED --add-opens=java.desktop/com.sun.awt=ALL-UNNAMED --add-opens=java.desktop/sun.awt.X11=ALL-UNNAMED &
```

The screenshot shows the 'Vitis IDE Launcher' window. It displays the command being run and a configuration dialog. The dialog has a title 'Select a directory as workspace'. It contains a text input field 'Workspace' with the value '/home/ubuntu/workspace', a 'Browse...' button, and a checkbox 'Use this as the default and do not ask again'. Below the input field are two sections: 'Restore other Workspaces' and 'Recent Workspaces', both currently empty. At the bottom are 'Cancel' and 'Launch' buttons.

- vitis_hls

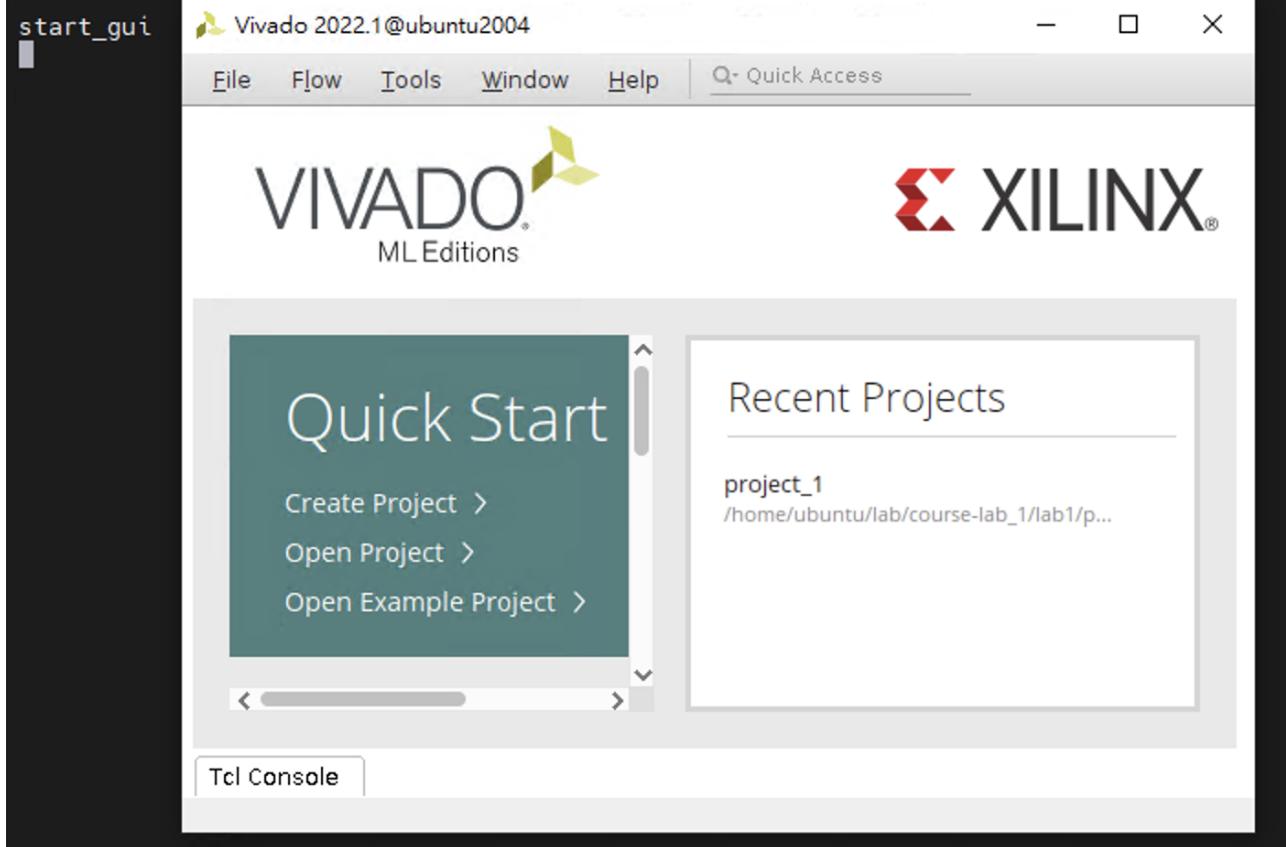
```
ubuntu@ubuntu2004:~$ vitis_hls
***** Vitis HLS - High-Level Synthesis from C, C++ and OpenCL v2022.1 (64-bit)
**** SW Build 3526262 on Mon Apr 18 15:47:01 MDT 2022
**** IP Build 3524634 on Mon Apr 18 20:55:01 MDT 2022
** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

source /tools/Xilinx/Vitis_HLS/2022.1/scripts/vitis_hls/hls.tcl -notrace
INFO: [HLS 200-10] Running '/tools/Xilinx/Vitis_HLS/2022.1/bin/unwrapped/lnx64.o/vitis_hls'
INFO: [HLS 200-10] For user 'ubuntu' on host 'ubuntu2004.linuxvmmimages.local' (Linux_x86_64 version 5.15.0-84-generic) on Thu Sep 28 06:14:53 EDT 2023
INFO: [HLS 200-10] On os Ubuntu 20.04.4 LTS
INFO: [HLS 200-10] In directory '/home/ubuntu'
INFO: [HLS 200-10] Bringing up Vitis HLS GUI ...
```

The screenshot shows the 'Vitis HLS 2022.1 @ubuntu2004' window. The menu bar includes File, Edit, Project, Solution, Window, and Help. A sub-menu 'Vitis HLS Welcome Page' is open, showing a sidebar with 'OPEN RECE' and a list of paths: 'lab1', '/home/ubuntu/lab', '/course-lab_1', and '/lab1'. The main area features the XILINX logo, the word 'VITIS', and a 'PROJECT' section with a 'Create Project' button.

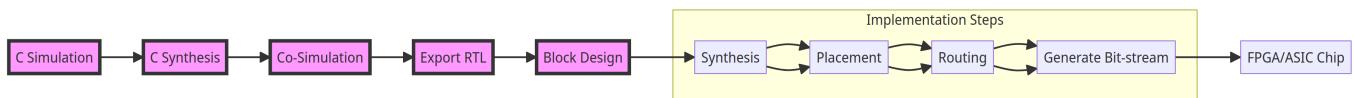
- vivado

```
ubuntu@ubuntu2004:~$ vivado
***** 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.
```



Introduction about the overall system

從實作 Lab1 過程中，完成跑過整個簡單的實驗流程，可將個階段步驟整合成以下流程：



Step

1. C Simulation:
 - 利用 C/C++ 撰寫欲達成目標邏輯算法 → 運行C Simulation工具 → 驗證程式碼正確性
2. C Synthesis:
 - 選擇合成設置 → 運行合成(Synthesis)工具 → 產生合成後的 report
3. Co-Simulation:
 - 使用C模擬的輸入/輸出 → 運行合成的HDL模擬 → 比較結果
4. Export RTL:
 - 從HLS工具中選擇輸出或導出RTL → 獲得HDL文件
5. Implementation:

在這個階段，HDL 被轉換成 FPGA 或 ASIC 的實際物理結構。選擇特定的 FPGA/ASIC → 運行 Implementation

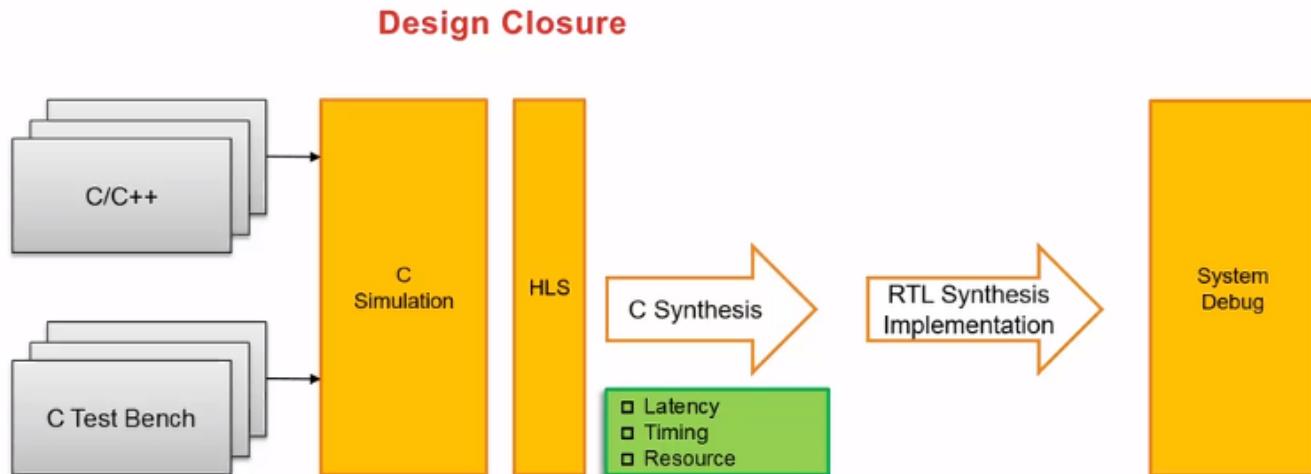
1. Synthesis (within Implementation):
 - 將HDL 轉換為一個 Netlist，這個 Netlist 包含邏輯閘、Flip-flop等之間的連接
2. Placement:
 - 決定 Netlist 中，每個元件在 FPGA 或 ASIC 物理布局上的位置
3. Routing:
 - 確定如何在 FPGA 或 ASIC 中連接這些元件，以形成完整的設計
4. Generate Bit-stream:
 - 創建一個代表整個 FPGA 配置的文件，可被下載到 FPGA 上執行

而 "Export RTL" 後，可進行 Block Design (系統集成)，使用圖形化介面連接不同的 IP 模組，也可以包含從 HLS 或其他來源導入的自定義 IP (Intellectual Property)。

系統概述

Lab1 使用的工具為 Vitis-HLS (Vitis High-Level Synthesis)，為基於 C++ 的開發工具，可以高階語言轉換為 RTL語言。

使用此設計流程，開發者可直接利用 C/C++ 來描述演算法及 Test Bench，執行 C Simulation 驗證設計的正確性。接著，透過高階合成工具，完成 C/C++ 轉換到 RTL。同時也需要考慮延時、時序以及資源的問題。然後，依照 RTL 設計流程，使用軟體工具進行 RTL 電路的合成、佈局以及佈線。最終，在系統層面上對整體設計進行驗證和修正。



Ref: <https://xilinx.eetrend.com/blog/2022/100560242.html>

Screen dump

- Performance

C Synthesis

Synthesis Summary Report of 'multip_2num'

General Information

Date: Thu Sep 28 03:13:38 2023
Version: 2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)
Project: lab1

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

| Target | Estimated | Uncertainty |
|----------|-----------|-------------|
| 10.00 ns | 6.912 ns | 2.70 ns |

Performance & Resource Estimates

| Modules & Loops | Issue Type | Violation Type | Distance | Slack | Latency(cycles) | Latency(ns) | Iteration Latency | Interval | Trip Count | Pipelined | BRAM | DSP | FF | LUT | URAM |
|-----------------|------------|----------------|----------|-------|-----------------|-------------|-------------------|----------|------------|-----------|------|-----|-----|-----|------|
| multip_2num | - | - | - | 3 | 30.000 | - | 4 | - | no | 0 | 3 | 409 | 307 | 0 | 0 |

Vitis HLS Console

```

INFO: [BLND 205-100] Finished micro-architecture generation.
INFO: [HLS 200-111] Finished Binding: CPU user time: 0.03 seconds. CPU system time: 0.02 seconds. Elapsed time: 0.06 seconds; current allocated memory: 483.164 MB.
INFO: [HLS 200-100] .....
INFO: [HLS 200-100] ... Generating RTL for module 'multip_2num'
INFO: [HLS 200-100] .....
WARNING: [RTGEN 206-101] Design contains AXI ports. Reset is fixed to synchronous and active low.
INFO: [RTGEN 206-500] Setting interface mode on port 'multip_2num/n32In1' to 's_axilite & ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on port 'multip_2num/n32In2' to 's_axilite & ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on port 'multip_2num/pn32ResOut' to 's_axilite & ap_vld'.
INFO: [RTGEN 206-500] Setting interface mode on function 'multip_2num' to 'ap_ctrl_none'.
INFO: [RTGEN 206-100] Bundling port 'n32In1', 'n32In2' and 'pn32ResOut' to AXI-Lite port control.
INFO: [RTGEN 206-100] Generating core module 'mul_32s_32_32_2_1'; 1 instance(s).
INFO: [RTGEN 206-100] Finished creating RTL model for 'multip_2num'.
INFO: [HLS 200-111] Finished creating RTL model: CPU user time: 0.01 seconds. CPU system time: 0.02 seconds. Elapsed time: 0.04 seconds; current allocated memory: 483.164 MB.
INFO: [HLS 200-111] Finishing Generation of all HLS modules: CPU user time: 0.14 seconds. CPU system time: 0.1 seconds. Elapsed time: 0.07 seconds; current allocated memory: 483.164 MB.
INFO: [VHDL 208-304] Generating VHDL RTL for multip_2num.
INFO: [VHDL 208-304] Generating Verilog RTL for multip_2num.
INFO: [VLOG 209-307] Generating Verilog RTL for multip_2num.
INFO: [HLS 200-789] **** Estimated Fmax: 144.68 MHz
INFO: [HLS 200-111] Finished Command csynth_design CPU user time: 5.1 seconds. CPU system time: 1.53 seconds. Elapsed time: 6.67 seconds; current allocated memory: 21.738 MB.
INFO: [HLS 200-112] Total CPU user time: 0.51 seconds. Total CPU system time: 4.76 seconds. Total elapsed time: 11.33 seconds; peak allocated memory: 484.238 MB.
Finished C synthesis.

```

Co-Simulation

Cosimulation Report for 'multip_2num'

General Information

Date: Thu 28 Sep 2023 02:04:12 AM EDT
Version: 2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)
Project: lab1
Status: Pass

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Cosim Options

Tool: Vivado XSIM
Dump Trace: all
RTL: Verilog

Performance Estimates

| Modules & Loops | Avg II | Max II | Min II | Avg Latency | Max Latency | Min Latency |
|-----------------|--------|--------|--------|-------------|-------------|-------------|
| multip_2num | 24 | 28 | 24 | 3 | 3 | 3 |

csynth.rpt

Performance & Resource Estimates:

```

PS: '+' for module; 'o' for loop; '*' for dataflow
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Modules | Issue |       | Latency | Latency| Iteration|       | Trip |       |       |
| & Loops | Type  | Slack| (cycles)| (ns)  |       | Interval| Count| Pipelined| BRAM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|+ multip_2num |      -| 0.39|       3| 30.000|      -|       4|      -|      no|      -
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

multip_2num_csynth.rpt**== Performance Estimates****+ Timing:***** Summary:**

| Clock | Target | Estimated | Uncertainty |
|---------|----------|-----------|-------------|
| lap_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 |

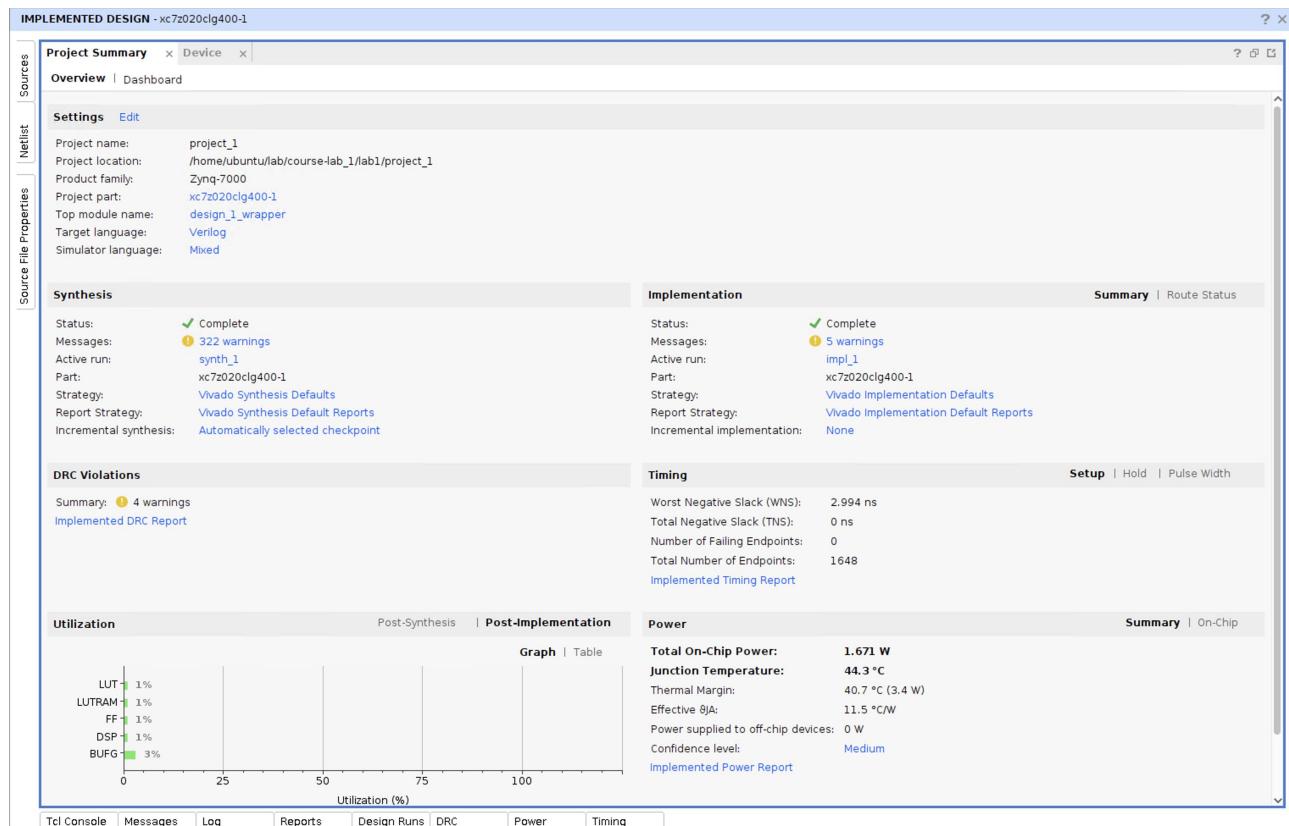
+ Detail:*** Instance:**

N/A

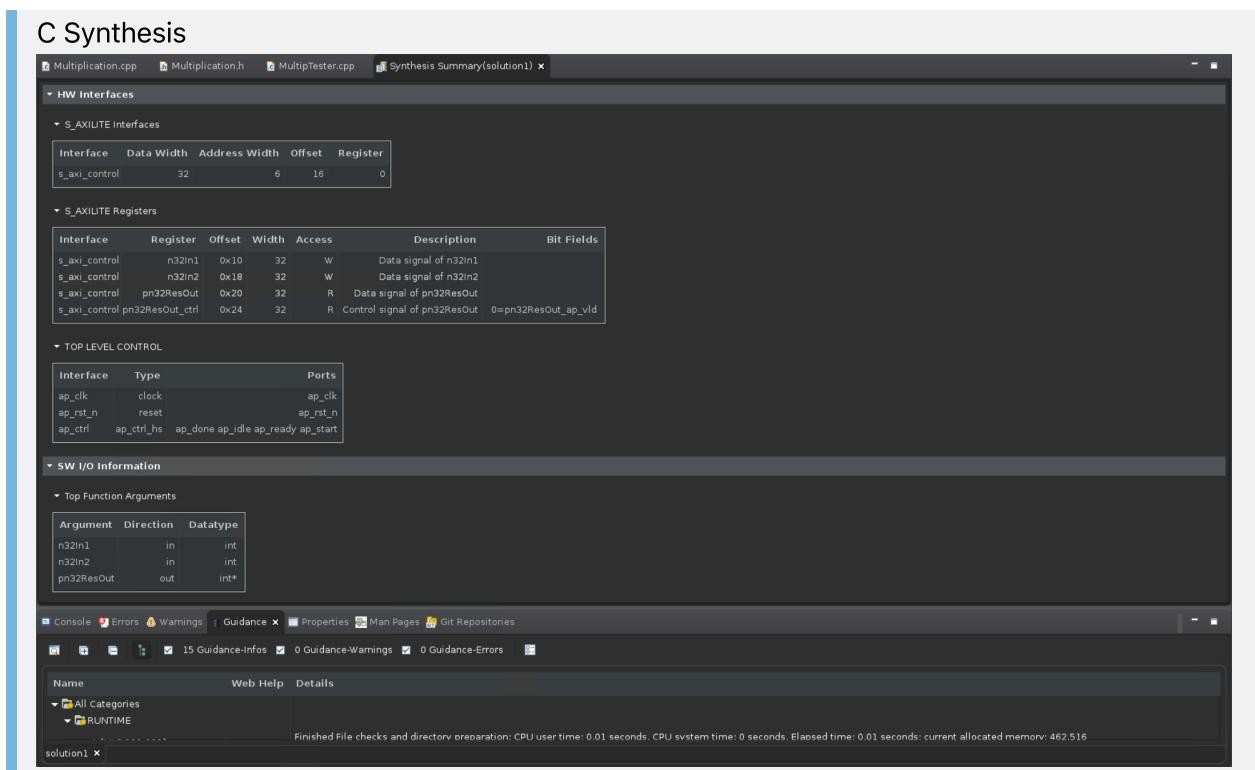
*** Loop:**

N/A

- Utilization



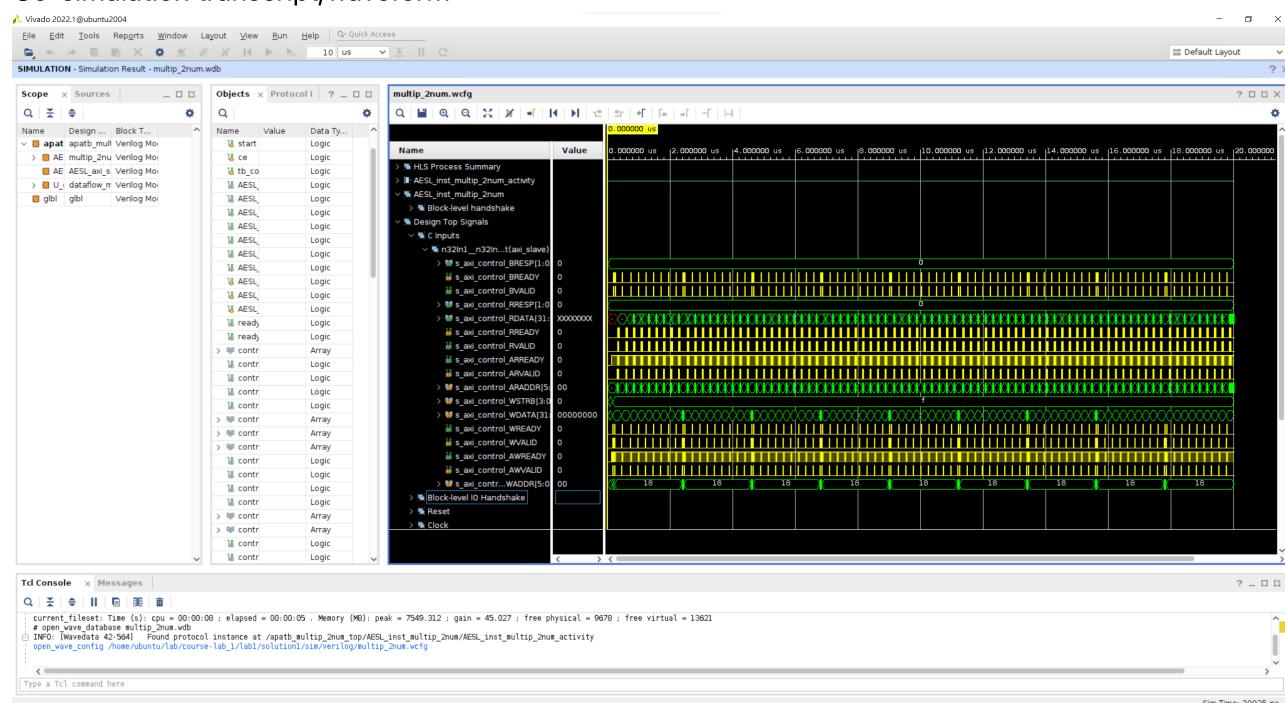
- Interface



multip_2num_csynth.rpt

| 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_none multip_2num return value | | | | |
| ap_rst_n | in 1 ap_ctrl_none multip_2num return value | | | | |

- Co-simulation transcript/waveform



- Jupyter Notebook execution result

The screenshot shows a Jupyter Notebook interface with a file browser. At the top, there's a navigation bar with tabs for 'Files', 'Running', and 'Clusters'. Below the navigation bar is a toolbar with buttons for 'Duplicate', 'Move', 'Download', 'View', 'Edit', 'Upload', 'New', and a refresh icon. The main area displays a list of files and directories:

| | Name | Last Modified | File size |
|-------------------------------------|-----------------------|---------------|-----------|
| <input type="checkbox"/> | common | 1年前 | |
| <input type="checkbox"/> | getting_started | 1年前 | |
| <input type="checkbox"/> | logictools | 1年前 | |
| <input type="checkbox"/> | pynq_composable | 1年前 | |
| <input type="checkbox"/> | pynq_peripherals | 2年前 | |
| <input type="checkbox"/> | SoC_Lab | 27分鐘前 | |
| <input checked="" type="checkbox"/> | Multip2Num.ipynb | 幾秒前 | 7.91 kB |
| <input type="checkbox"/> | Welcome to Pynq.ipynb | 1年前 | 1.89 kB |
| <input checked="" type="checkbox"/> | Multip2Num.bit | 20分鐘前 | 4.05 MB |
| <input checked="" type="checkbox"/> | Multip2Num.hwh | 20分鐘前 | 150 kB |

```
In [1]: # coding: utf-8
# In[1]:

from __future__ import print_function
import sys, os
sys.path.append('/home/xilinx')
os.environ['XILINX_XRT'] = '/usr'
from pynq import Overlay
if __name__ == "__main__":
    print("Entry:", sys.argv[0])
    print("System argument(s):", len(sys.argv))
    print("Start of " + sys.argv[0] + "\n")
    ol = Overlay("/home/xilinx/jupyter_notebooks/Multip2Num.bit")
    regIP = ol.multip_2num_0
    for i in range(9):
        print("*****")
        for j in range(9):
            regIP.write(0x10, i + 1)
            regIP.write(0x10, j + 1)
            Reg = regIP.read(0x20)
            print(str(i + 1) + " * " + str(j + 1) + " = " + str(Reg))
        print("*****")
    print("Exit process")
```

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
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

Exit process

由於截圖範圍大，因此解析度有點損失

Screen dump (Other)

- C Synthesis

The screenshot shows the Vivado IP Flow interface with the following details:

- General Information:**
 - Date: Thu Sep 28 01:52:33 2023
 - Version: 2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)
 - Project: lab1
 - Solution: solution1 (Vivado IP Flow Target)
 - Product family: zynq
 - Target device: xc7z020-clg400-1
- Timing Estimate:**

| Target | Estimated | Uncertainty |
|----------|-----------|-------------|
| 10.00 ns | 6.912 ns | 2.70 ns |
- Performance & Resource Estimates:**

| Modules & Loops | Issue Type | Violation Type | Distance | Slack | Latency(cycles) | Latency(ns) | Iteration Latency | Interval | Trip Count | Pipelined | BRAM | DSP | FF | LUT | URAM |
|-----------------|------------|----------------|----------|-------|-----------------|-------------|-------------------|----------|------------|-----------|------|-----|-----|-----|------|
| multip_2num | - | - | - | 3 | 30.000 | - | - | 4 | - | no | 0 | 3 | 409 | 307 | 0 |
- HW Interfaces:**
 - S_AXILITE Interfaces:**

| Interface | Data Width | Address Width | Offset | Register |
|---------------|------------|---------------|--------|----------|
| s_axi_control | 32 | 6 | 16 | 0 |
- Console:** Shows 15 Guidance-Infos, 0 Guidance-Warnings, and 0 Guidance-Errors.

This screenshot shows the same Vivado IP Flow interface as above, but with expanded sections:

- HW Interfaces:**
 - S_AXILITE Interfaces:**

| Interface | Data Width | Address Width | Offset | Register |
|---------------|------------|---------------|--------|----------|
| s_axi_control | 32 | 6 | 16 | 0 |
 - S_AXILITE Registers:**

| 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 |
 - TOP LEVEL CONTROL:**

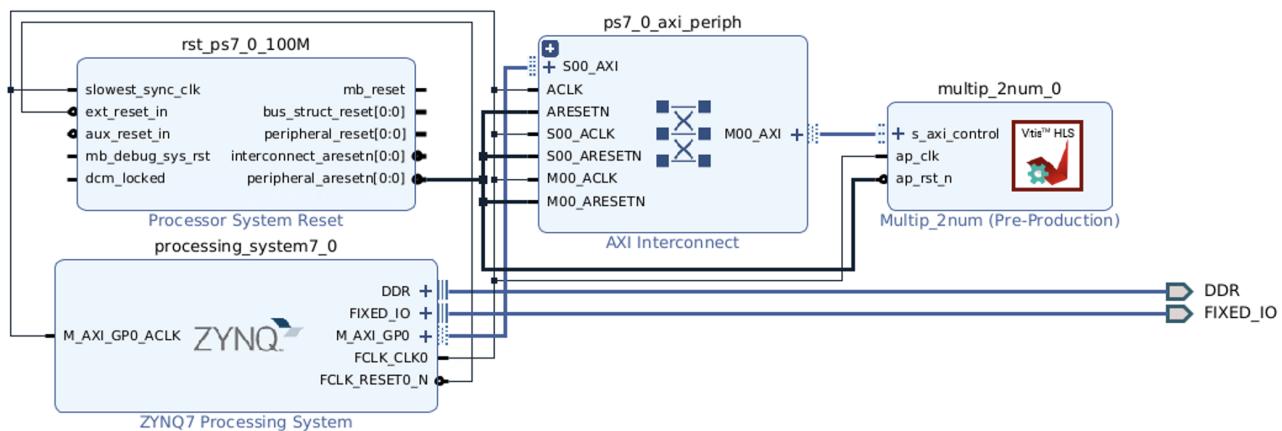
| Interface | Type | Ports |
|-----------|------------|-----------------------------------|
| ap_clk | clock | ap_clk |
| ap_rst_n | reset | ap_rst_n |
| ap_ctrl | ap_ctrl_hs | ap_done ap_idle ap_ready ap_start |
- SW I/O Information:**
 - Top Function Arguments:**

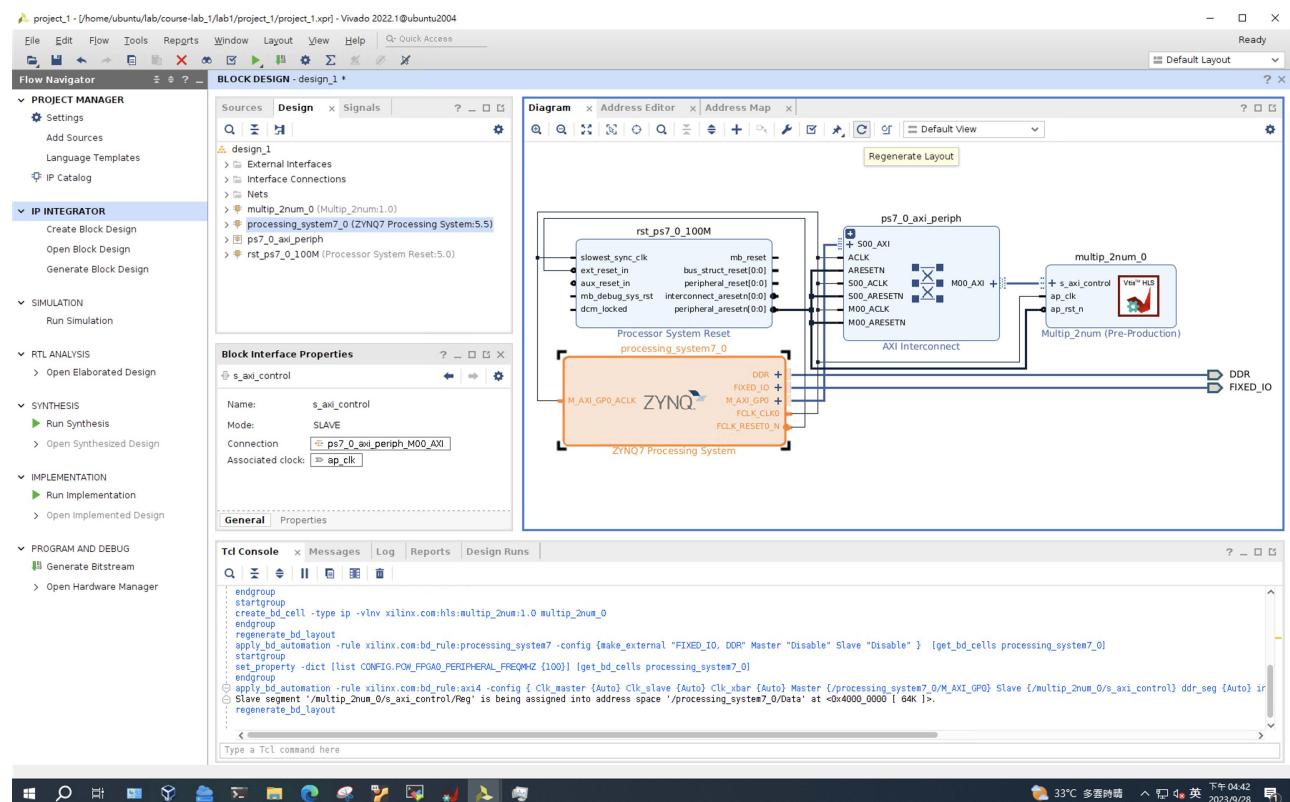
| Argument | Direction | Datatype |
|------------|-----------|----------|
| n32in1 | in | int |
| n32in2 | in | int |
| pn32ResOut | out | int* |
- Console:** Shows 15 Guidance-Infos, 0 Guidance-Warnings, and 0 Guidance-Errors.

The screenshot shows the Vivado IDE interface with the following sections visible:

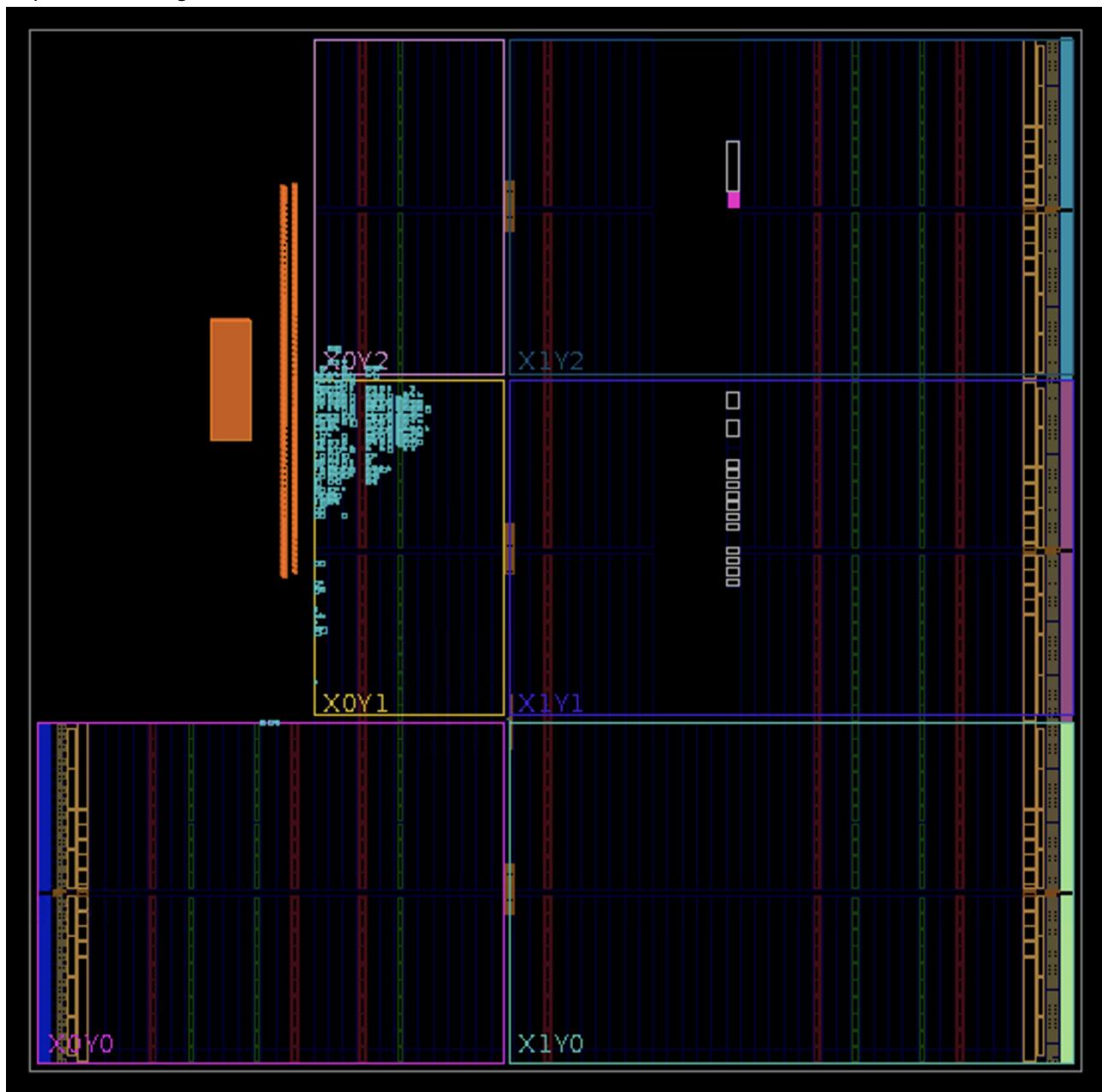
- SW I/O Information:** Shows function arguments and their mappings to hardware.
- HW Info:** Shows the hardware interface for each argument.
- Pragma Report:** Shows valid pragma syntax and bind op report.
- User config_op:** Shows configuration options for the multip_2num component.
- Console:** Displays build logs and error messages.
- Properties:** Shows runtime properties for the project.

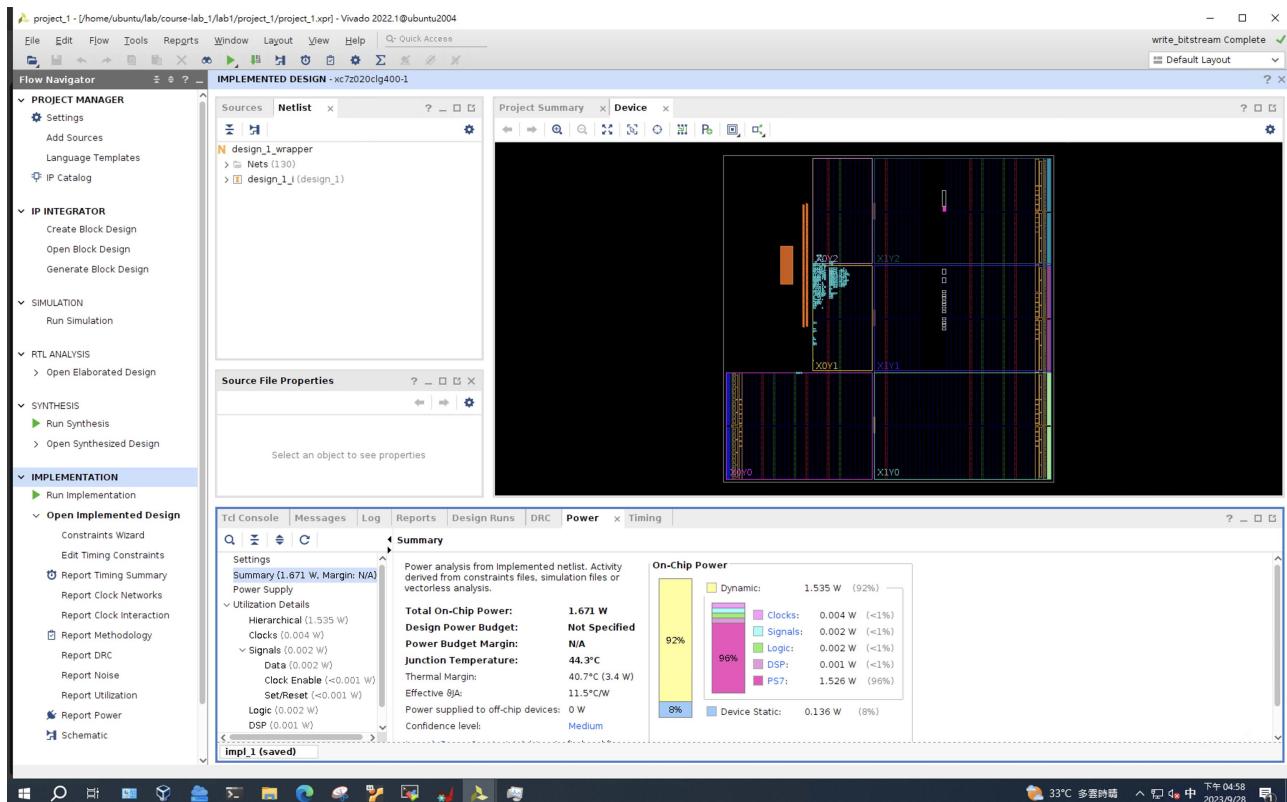
- Block Design





- Implement Design





Note

名詞解釋

• C Simulation:

- 目的: 驗證C/C++的算法邏輯正確性
- 說明: 為初步的模擬階段，只涉及C/C++程式，不涉及任何硬體描述語言。基本上，這階段確保C/C++程式在未進行任何合成或硬體轉換的情況下運行正確性
- 結果: 若發現錯誤，需要於這階段修改C/C++程式

• C Synthesis:

- 目的: 將轉換C/C++到硬體描述語言 (HDL)，如 VHDL 或 Verilog。
- 說明: 高階合成 (HLS) 工具會嘗試將C/C++程式碼轉化成硬體結構。在這階段，可能需要進行優化或改善，以達到所需的clock速度、資源使用或功耗
- 結果: 得到合成後的報告(`csynth.rpt`)，包括估計的資源使用、性能(Performance)等

• Co-Simulation:

- 目的: 驗證合成後的HDL程式碼與C/C++原始碼的行為是否一致。
- 說明: 在這階段，原始C/C++程式碼和其合成的HDL程式碼將同時模擬。這是確保合成後的HDL實現與原始演算邏輯在功能上保持一致
- 結果: 若發現差異或問題，需回到C Synthesis階段，調整優化指令或更改C/C++程式，然後重新合成和模擬

• Export RTL:

- 目的: 生成代表特定設計的硬體描述語言 (HDL)，如 VHDL 或 Verilog。
- 說明: 在完成高階合成 (HLS) 後，已從其C/C++ 轉換為硬體結構。而 "Export RTL" 將硬體結構生成為可用於標準 FPGA 或 ASIC toolchain 的 HDL 程式碼的過程
- 結果: 生成的 HDL 可以被整合到更大的系統中，或者直接使用在 FPGA 或 ASIC 設計中

• IP (Intellectual Property):

- 目的: 封裝特定的設計或功能以便重複使用

- 說明: IP 是指一個已經設計好的、可以重複使用的功能或子系統。IP 可為自己設計，亦可為第三方供應商提供。當設計被合成並驗證無誤後，可被封裝成一個 IP，以便在其他設計中重複使用或共享
- 結果: 生成的 IP 模組通常以標準格式提供，如：Xilinx 的 AXI4 接口，允許被整合到更大的系統中

觀念補充

1. 在 HLS 中，"pragmas" 和 "directives" 通常指同樣的東西，都是用於指導合成過程的指令。允許優化和指導合成過程，而不必大量修改 C/C++ 的原始碼。
2. reg (register) 為存儲單元。在 FPGA 設計中，register 是用來保存狀態或資料的基本元件，由 **flip-flops** 構成。
 - **regIP = ol.multip_2num_0**，代表 IP 核心的物件，可通過 Python 存取的 register
 - 每個在 FPGA overlay 中的 IP 都可利用其名稱於 Python 存取和控制
3. **Overlay**: 將 bitstream 載入到 FPGA，並將相應的硬體設計在 FPGA 上執行
 - MMIO (Memory-Mapped I/O): 允許 CPU 使用常規的記憶體存取指令，如：load 和 store
 - 特定 I/O 裝置的 register 或控制位會映射(**mapping**)到系統的主記憶體地址空間中的某些位置
 - 在 PYNQ 環境中，利用 MMIO 直接訪問 FPGA 上的 MMIO 地址空間，能夠讀取與寫入 FPGA 中的硬體 register
 - 使用 Overlay 載入 FPGA 設計時，各個 IP 核心可能有屬於自己的 MMIO 地址範圍，允許從 Python 存取和控制這些核心

心得

由於為資工系的學生，之前並未有機會接觸到SoC的相關知識和設計。在這學期的SoC課程中，首次深入接觸SoC的設計與相關知識。於Lab1實作過程中，遇到了許多我不熟悉的專業名詞和實作原理。為了更好掌握這些新知識，因此投入時間學習和理解。而透過Lab1的初步實作更讓我體驗到工具的操作，如: vitis_hls，這都使我對SoC有了更深入的了解，從而加深了我的理解與實踐能力。

而本次實驗過程中，對於 OnlineFPGA 的租借機制與 infra 感到非常驚艷，利用所提供的平台服務，可快速建立驗證環境，並且可不侷限於理設備的實體實驗限制與配置的複雜性，利用遠端方式就可以快速完成 FPAG 的設計與驗證以及測試。

未來預期目標

- 目前可能有些名詞用語不夠精準，期望後續能逐漸了解其意義，並正確的使用名詞說明實作過程的體悟與想法

Troubleshooting

- None

補充

後續更多資訊會補充於以下HackMD

- <https://hackmd.io/@Sheng08/HkzeBuDyp>

 **TODO**

- 了解 ASIC
- 了解 正反器(Flip-flop) 與查找表(LUT)