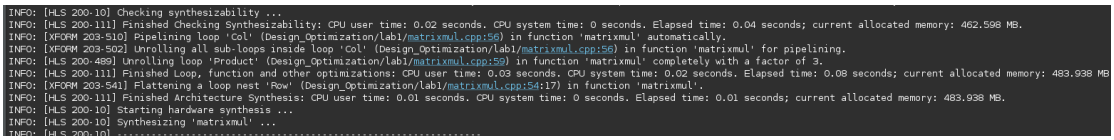
Introduction

在 Lab A 我是選 Chapter 7 Design Optimization，此章節是用 matrix multiplications 當作例子，透過不同的 directive 設定讓整體 performance 上升，在這個 Chapter 有兩個 lab，總共有 6 個 solutions 一步一步的優化整個電路。

Lab 1 Optimizing a Matrix Multiplier

- Experimental Steps
  - Step 1 Create and Open the Project
  - Step 2 Synthesize and Analyze the Design

    一開始直接跑 synthesis 從 console 那邊得出他自動優化了哪些步驟，



    並得到合成完的 report

matrixmul.cpp | Synthesis Summary(solution1) | Synthesis Details(solution1)(matrixmul_csynth.rpt)

Project: matrixmul
Solution: solution1 (Vivado IP Flow Target)
Product family: virtexuplus
Target device: xcu50-fsvh2104-2-e

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	13.33 ns	2.870 ns	3.60 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
23	23	0.307 us	0.307 us	24	24	no

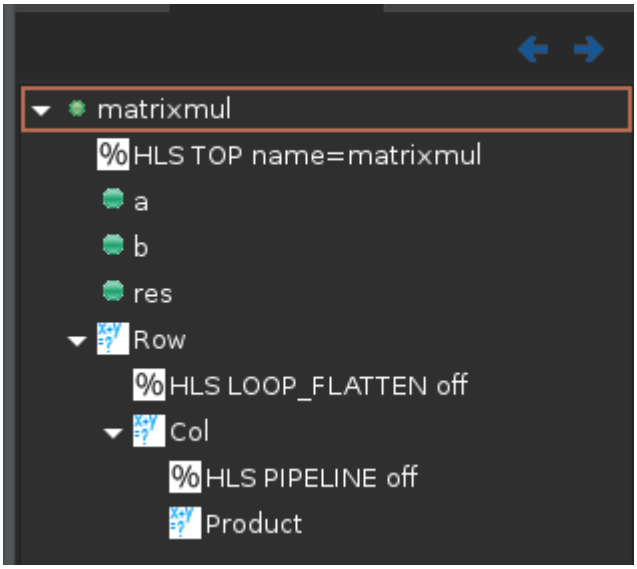
Detail

Instance

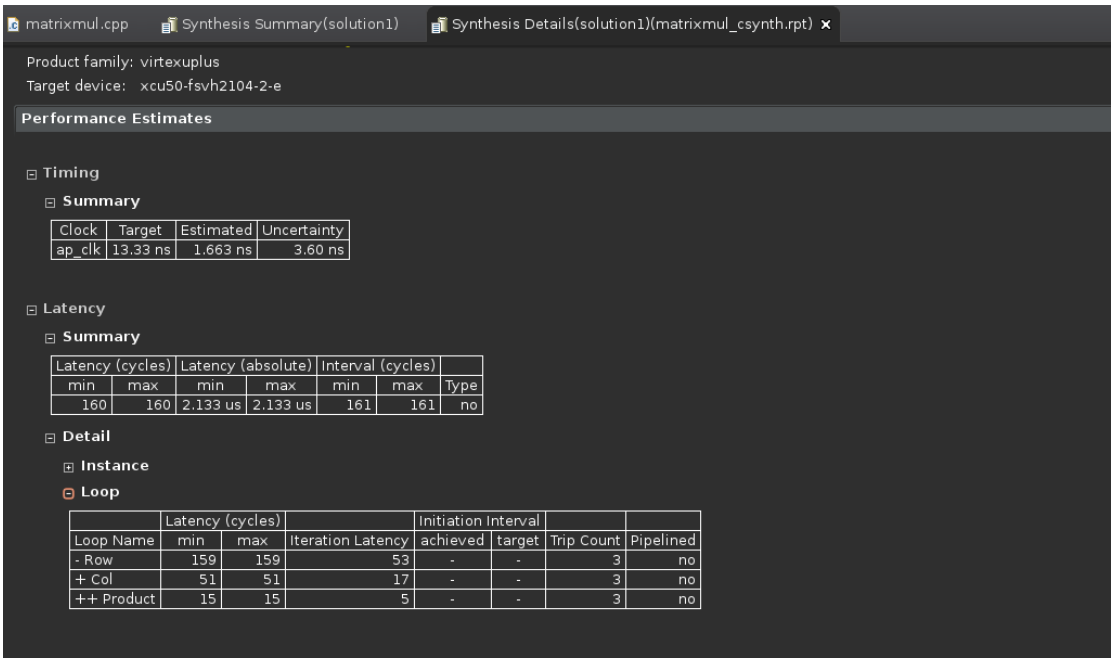
Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Row_Col	21	21	6	2	1	9	yes

於是我透過以下 directive 的設定盡量還原到沒有優化過的樣子，



而下圖是合成完的 report


matrixmul.cpp Synthesis Summary(solution1) Synthesis Details(solution1)(matrixmul_csynth.rpt)

Product family: virtexuplus
Target device: xcu50-fsvh2104-2-e

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	13.33 ns	1.663 ns	3.60 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
160	160	2.133 us	2.133 us	161	161	no

Detail

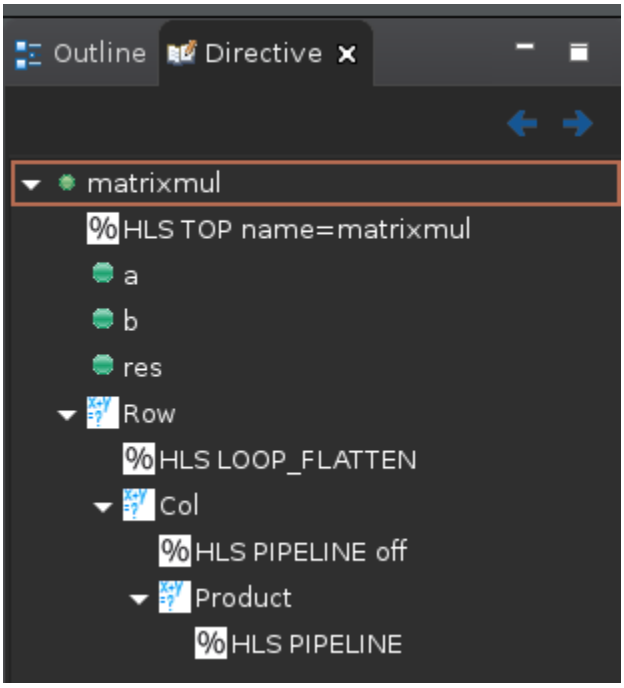
Instance

Loop

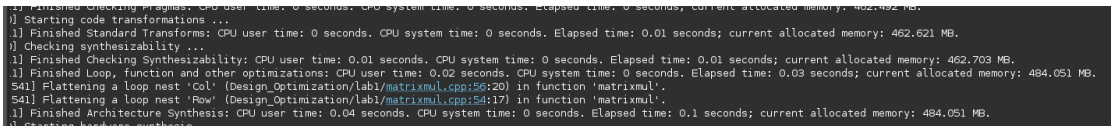
	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Row	159	159	53	-	-	3	no
+ Col	51	51	17	-	-	3	no
++ Product	15	15	5	-	-	3	no

■ Step 3 Pipeline the Product Loop

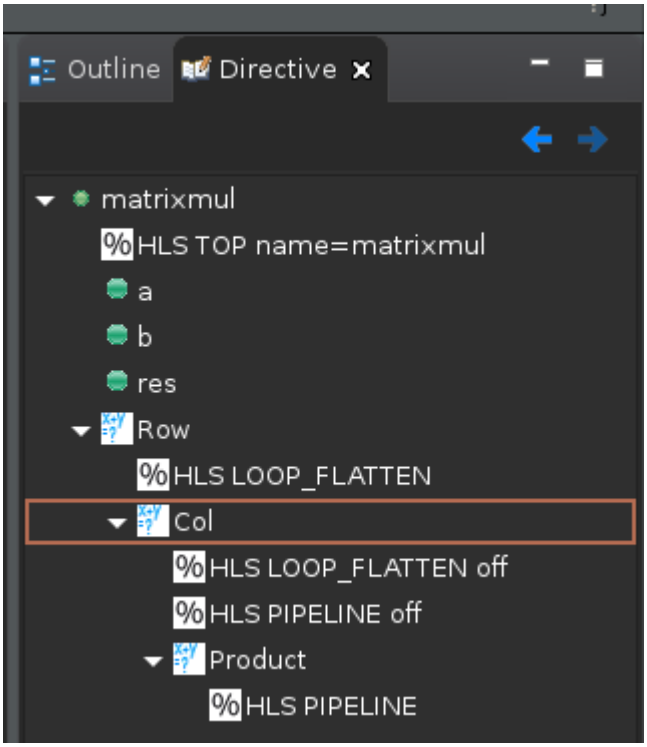
以下是 sol2 的 directive



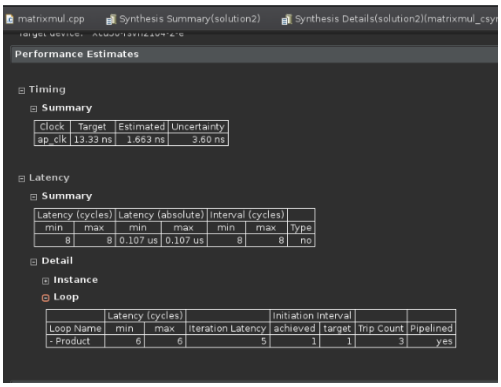
而他在合成時 console 出現以下資訊，

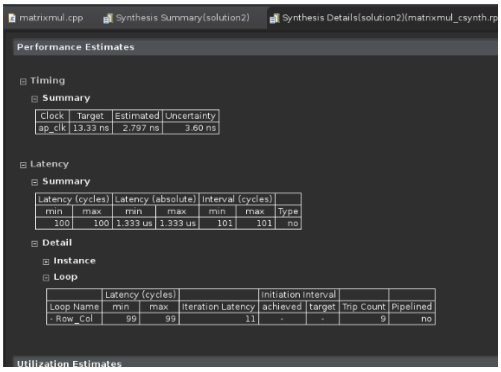


於是我將 directive 稍微修改成和 tutorial 相同的情況，



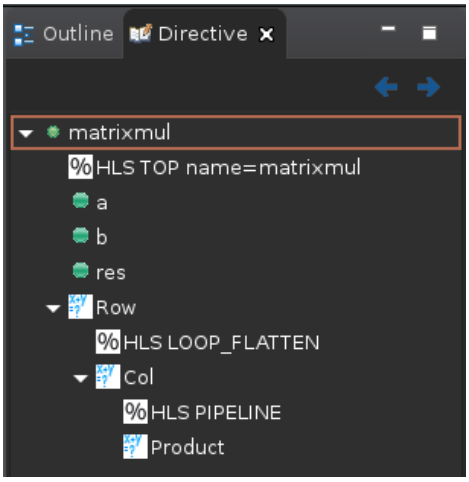
下圖是 sol 2 的合成 report



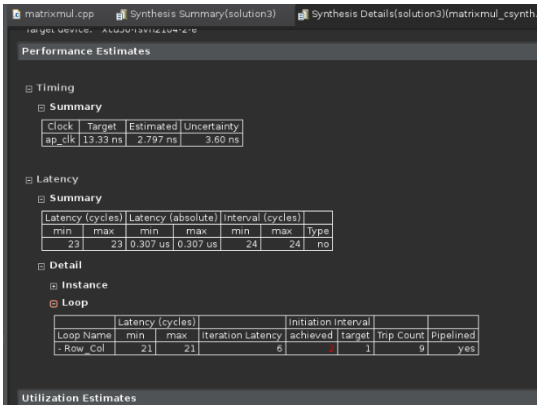


■ Step 4 Pipeline the Col Loop

下圖是 sol 3 的 directive 設定

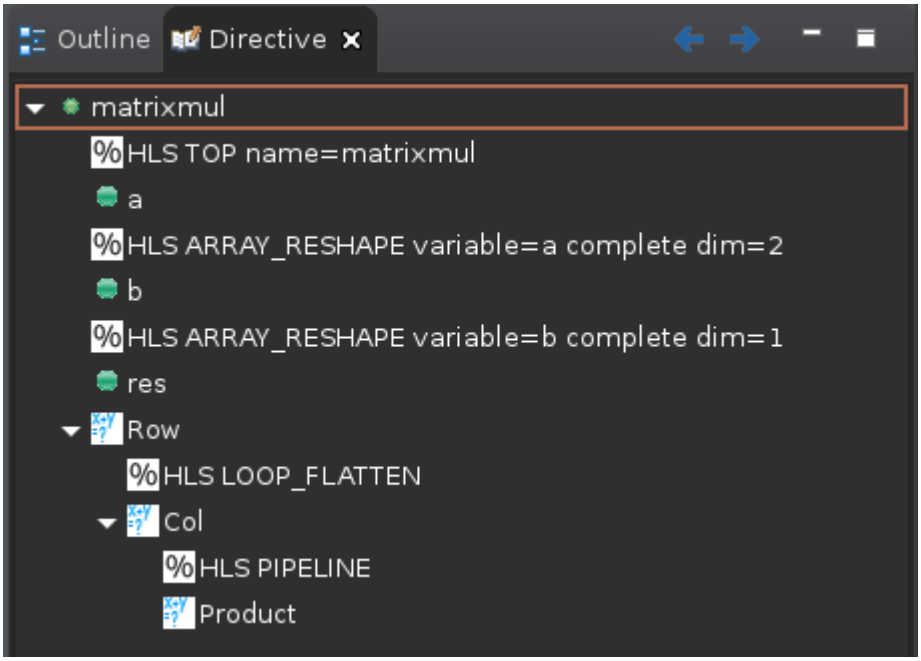


下圖是合成後的 report

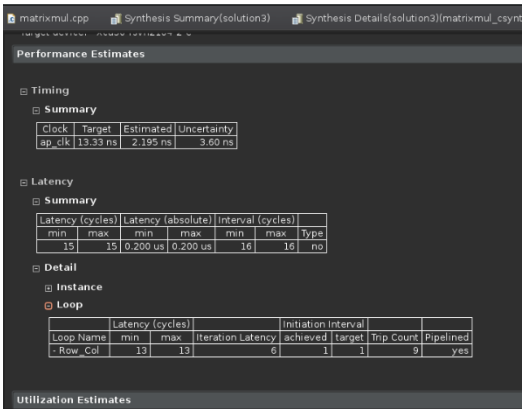


■ Step 5 Reshape the Arrays

下圖是 sol 4 的 directive

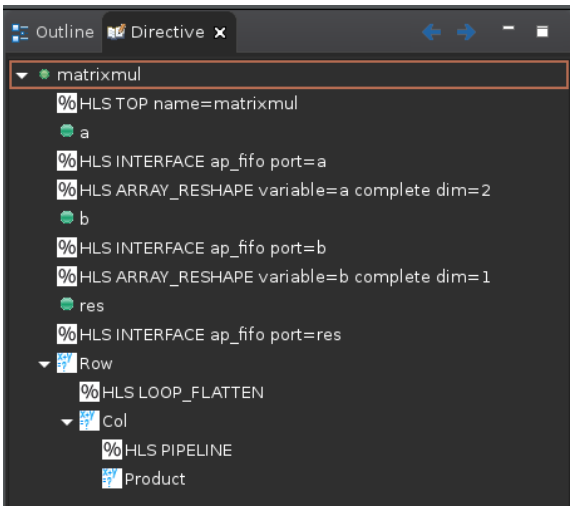


下圖是合成後的 report

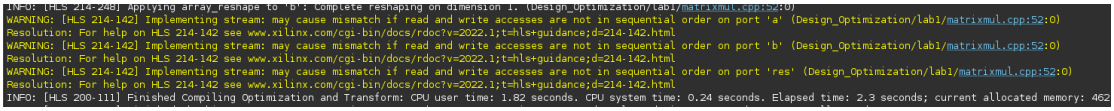


■ Step 6 Apply FIFO Interfaces

下圖是 sol 5 的 directive



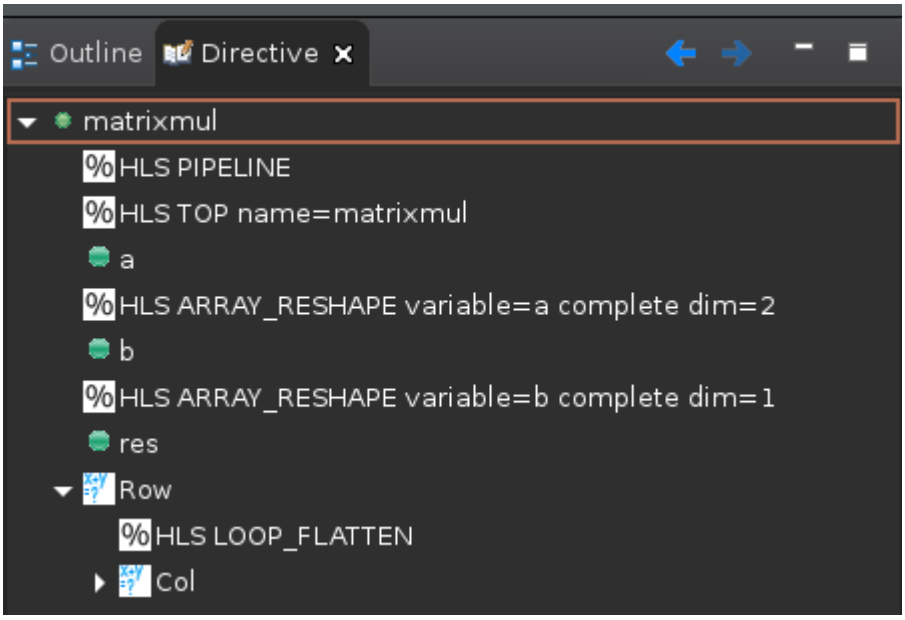
但是在合成的時候有出現 Warning，如下圖



得知無法用 fifo interface，需要修改 source code。

■ Step 7 Pipeline the Function

最後是 sol 6 的 directive



下圖是 sol 4 和 sol 6 的比較

matrixmul.cpp　Synthesis Summary(solution6)

solution4: xcu50-fsvh2104-2-e

solution6: xcu50-fsvh2104-2-e

Performance Estimates

Timing

Clock		solution4	solution6
ap_clk	Target	13.33 ns	13.33 ns
	Estimated	2.195 ns	2.249 ns

Latency

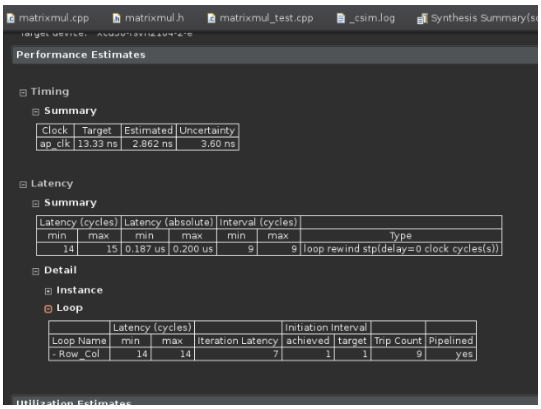
		solution4	solution6
Latency (cycles)	min	15	10
	max	15	10
Latency (absolute)	min	0.200 us	0.133 us
	max	0.200 us	0.133 us
Interval (cycles)	min	16	5
	max	16	5

Utilization Estimates

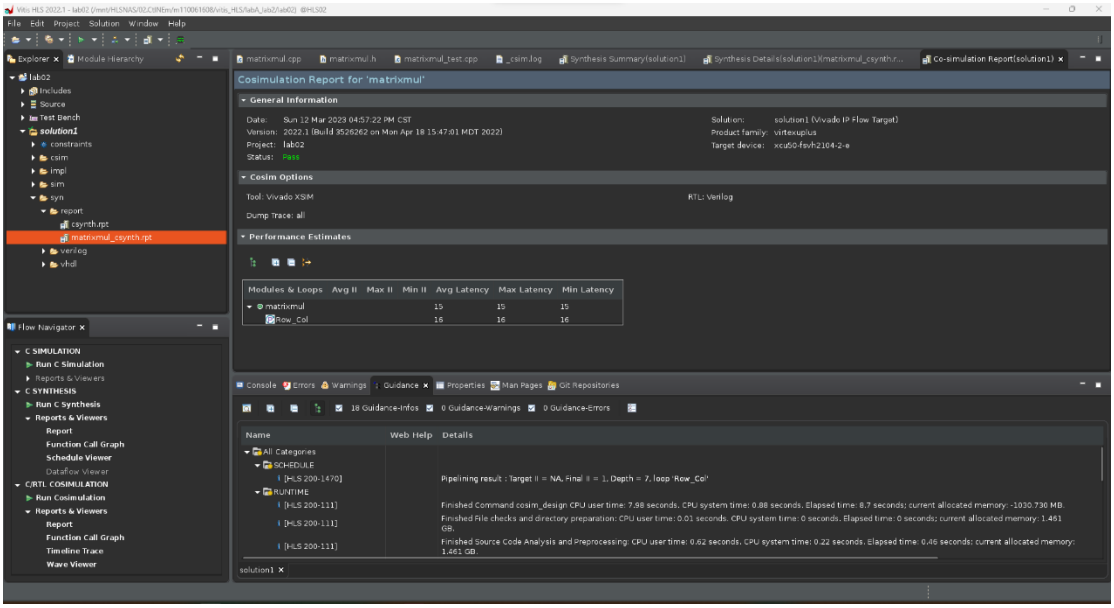
	solution4	solution6
BRAM_18K	0	0
DSP	2	18
FF	199	632
LUT	245	553
URAM	0	0

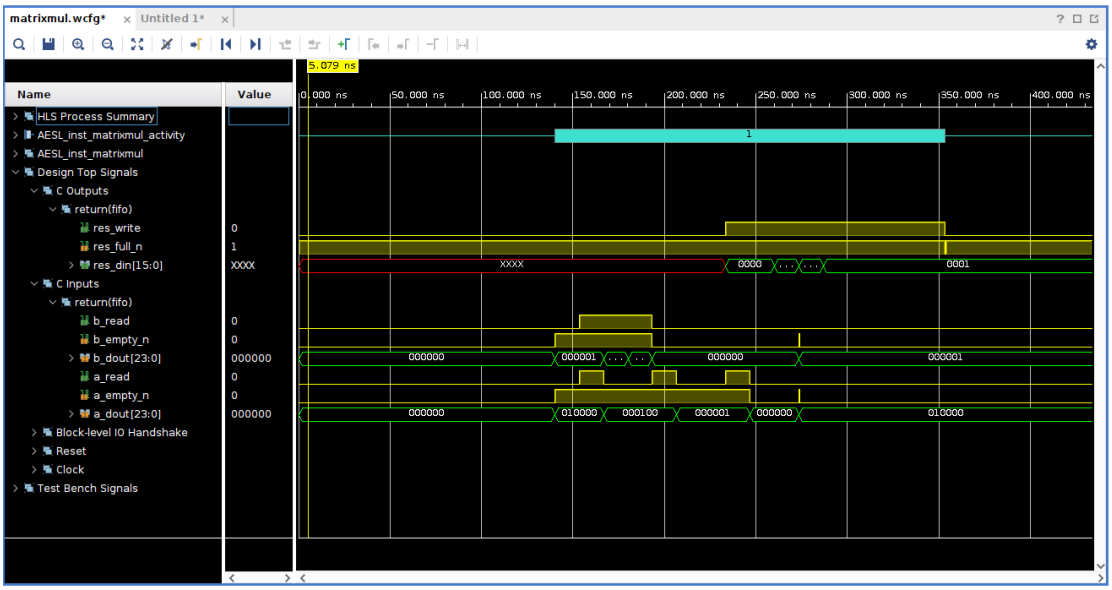
- Lab 2 C Code Optimized for I/O Accesses
  - Step 1 Create and Open the Project

下圖是合成後的 report



下圖是 co-sim 的 report 和 waveform





Github link ： https://github.com/sssh311318/HLS_LAB_A