NCTU-EE DCS-2021

HW03

Design: Matrix CPU with Pipeline

資料準備

1. 從 TA 目錄資料夾解壓縮

   % tar -xvf ~dcsta01/HW03.tar

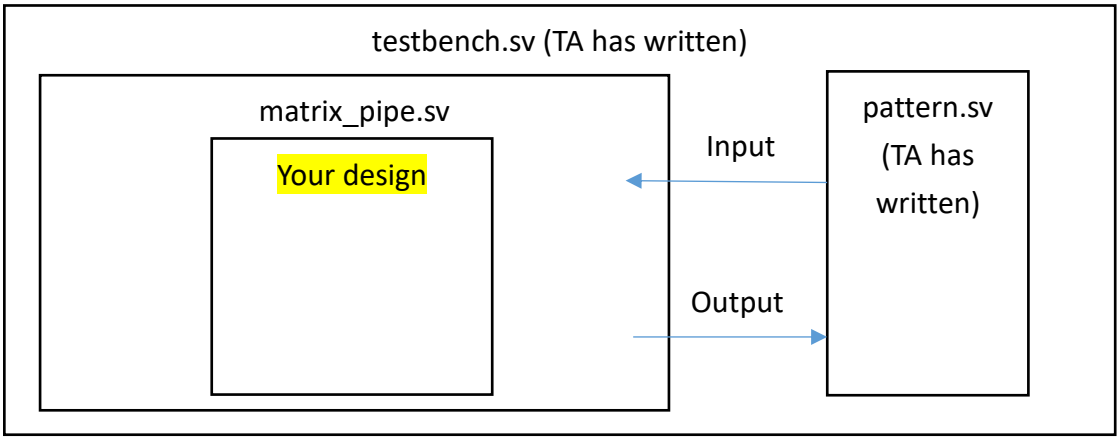
2. 解壓縮資料夾 HW03 包含以下：

   A. 00_TESTBED/

   B. 01_RTL/

   C. 02_SYN/

Block Diagram



設計描述

此次作業目的是設計一個簡單的矩陣運算單元，其中含有**兩階段運算**部分，分別為 **3x1 與 3x3**，而此次作業會運用到 Pipeline 技巧。

此次作業中資料的 input 有兩個，分別為 input_matrix1 與 input_matrix2，input_matrix1 為 36bit，input_matrix2 為 12bit，每 4bit 為矩陣中的一個數字(**有號數**)，因此 input_matrix1 有 9 個數字(3x3)，input_matrix2 有 3 個數字(3x1)，順序為以下。第 1 數為 input_matrix1[3:0]，依此類推。

$$\text{input\_matrix1} = \begin{bmatrix} \text{第 1 數} & \text{第 2 數} & \text{第 3 數} \\ \text{第 4 數} & \text{第 5 數} & \text{第 6 數} \\ \text{第 7 數} & \text{第 8 數} & \text{第 9 數} \end{bmatrix}$$

$$\text{input\_matrix2} = \begin{bmatrix} \text{第 1 數} \\ \text{第 2 數} \\ \text{第 3 數} \end{bmatrix}$$

第一階段運算:

為 3x3 矩陣乘 3x1 矩陣，假設矩陣如下。

$$\text{input\_matrix1} = \begin{bmatrix} -1 & 2 & 3 \\ -4 & 5 & 6 \\ -7 & 8 & 9 \end{bmatrix} \quad \text{input\_matrix2} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

相乘後的結果為

$$\text{result\_matrix} = \begin{bmatrix} 12 \\ 24 \\ 36 \end{bmatrix}$$

此為第一階段的運算

第二階段運算:

將第一階段的結果擴展為 3x3，並依據三數相加後大小分為兩種狀況，乘上固定的 3x3 矩陣。如以下範例。

fixed_matrix 的數值請直接寫在自己的 design 中即可。

第一種情況 ：若上一階段的結果三數(12+24+36)相加大於 0 則乘上 fixed_matrix1

$$\text{stage2\_matrix} = \begin{bmatrix} 12 & 12 & 12 \\ 24 & 24 & 24 \\ 36 & 36 & 36 \end{bmatrix} \quad \text{fixed\_matrix1} = \begin{bmatrix} 1 & 2 & 4 \\ -2 & -3 & 1 \\ 0 & 5 & 6 \end{bmatrix}$$

相乘後的結果為

$$\text{result\_matrix} = \begin{bmatrix} -12 & 48 & 132 \\ -24 & 96 & 264 \\ -36 & 144 & 396 \end{bmatrix}$$

第二種情況 ：若上一階段的結果三數(假設為 -1 -2 -8)相加小於 0 則乘上 fixed_matrix2

$$\text{stage2\_matrix} = \begin{bmatrix} -1 & -1 & -1 \\ -2 & -2 & -2 \\ -8 & -8 & -8 \end{bmatrix} \quad \text{fixed\_matrix2} = \begin{bmatrix} -2 & 5 & 6 \\ 1 & 2 & 0 \\ 6 & 7 & -7 \end{bmatrix}$$

相乘後的結果為

$$\text{result\_matrix} = \begin{bmatrix} -9 & -4 & 13 \\ -10 & -28 & 2 \\ -40 & -112 & 8 \end{bmatrix}$$

此為第二階段的運算，最終再將 result_matrix 依序輸出至 out_number1~9，順序依照左上至右下的順序(與第一頁 input_matrix1 順序所示相同)。

※注意事項:

1.此次作業的輸入訊號為連續輸入，不如以往作業 lab，因此需要使用 pipeline 技巧。

2.此次作業有大量乘法與加法，無法在一個 cycle 內進行全部計算，因此請自行設計 pipeline 架構、級數。

3.通常而言，越需要高速運算(1cycle 內進行大量運算)，面積則會越大，因此請自行斟酌設計。

Input

Signal name	Number of bit	Description
clk	**1 bit**	clock
rst_n	**1 bit**	Asynchronous active-low reset
in_valid	**1 bit**	When getting high, means start giving instruction
input_matrix1	**36 bit**	3x3 的矩陣，每 4bit 為一個數值，各數值為 signed
input_matrix2	**12 bits**	313 的矩陣，每 4bit 為一個數值，各數值為 signed

Output

Signal name	Numb	Description
out_valid	**1 bits**	當做完運算之後，為 high，代表 out_number1~9 有值，當運算結束後，須歸零。Invalid 後須在 50cycle 內有值。

out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 1 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 2 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 3 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 4 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 5 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 6 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 7 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 8 個數值。
out_number1	**16 bit**	Signed，兩階段運算後的矩陣的第 9 個數值。

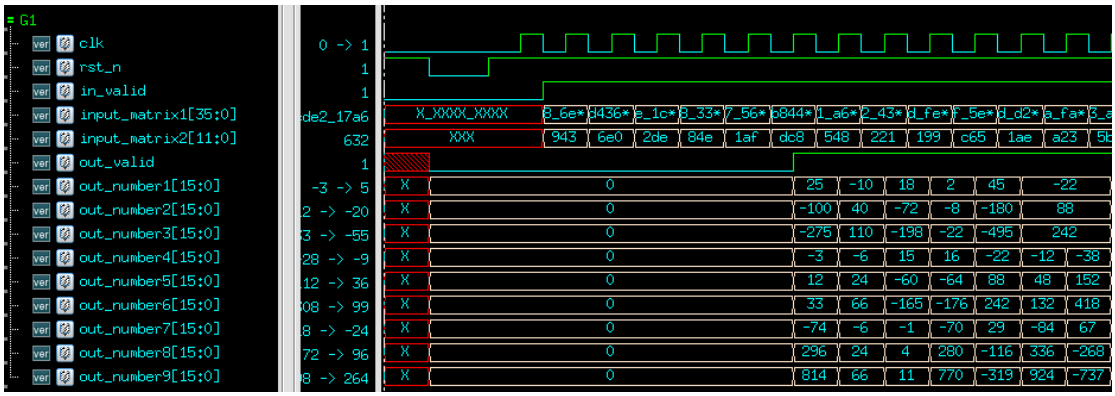
以上的 output 當 pattern 吐完 input、計算完後皆須歸為 0。

Specification

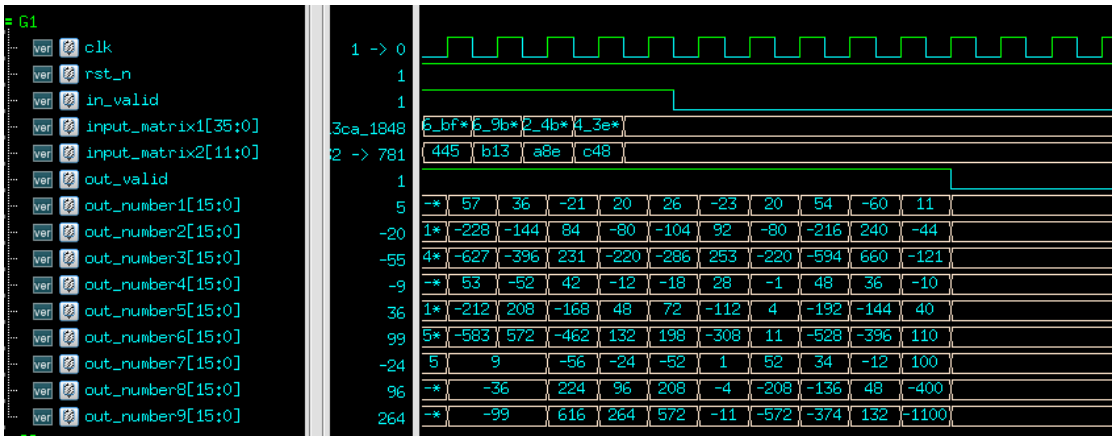
1. Top module name : matrix_pipe (File name: matrix_pipe.sv)
2. 請用 System verilog 完成你的作業。
3. 請使用 pipeline。
4. 02_SYN result 不行有 error 且不能有 latchs。
5. 02_SYN result slack time 需>=0
6. 需在 invalid 為 1 後 50Cycle 內 output 第一筆答案，並連續輸出的第 2 筆第 3 筆。
7. 運算結束後 output 須歸零。

Example Waveform

Pattern 一開始



Pattern 結束



上傳檔案

1. mips_pipe_dcsxx.sv (使用 09 上傳 會自動改檔名)
2. report_dcsxx.pdf, **xx is your server account.** (上傳至 E3)
3. 請 4/29 15:30 之前上傳 (補交期限為 5/6 15:30)

Grading Policy

1. Pass the RTL & Synthesis simulation. 70%
2. Area * Latency 20%
3. Report. 10%

Note

Template folders and reference commands:

1. 01_RTL/ (RTL simulation) -> ./01_run
2. 02_SYN/ (Synthesis) -> ./01_run_dc
3. 09_UPLOAD/(upload) -> ./01_upload

報告請簡單且重點撰寫，不超過兩頁 **A4**，並包括以下內容

1. 描述你的設計方法，包含但不限於如何加速**(**減少 **critical path)**或降低面積。
2. 基於以上，畫出你的架構圖**(Block diagram)**，**pipeline** 的架構等
3. 心得報告，不侷限於此次作業，對於作業或上課內容都可以寫下。