**國立清華大學 電機工程系**

**112學年度第一學期**

**SOC Design Laboratory**

**Lab #4-2**

**學校系所：清大電機所、電子所**

**學號姓名：112061611 陳伯丞**

**112061524 葉又菘**

**110063553 張傑閔**

**指導教授****：****賴瑾 教授 ( Prof. Jiin Lai )**

**中 華 民 國 一 一 二 年 十 一 月**

**Table of Contents**

**Table of content** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . i

**1 Block diagram** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.1 Datapath . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.2 Control-path . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

**2 Interface protocol** . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

**3 Waveform** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

**4 Questions**  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

**5 GitHub**  . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . 13

**Block diagram**

* 1. **Datapath**

一張含有 文字, 螢幕擷取畫面, 圖表, Rectangle 的圖片

自動產生的描述

Fig 1. Datapath diagram

一張含有 文字, 螢幕擷取畫面, 圖表, Rectangle 的圖片

自動產生的描述Fig 2. User project block of Datapath diagram

* 1. **Controls-path**

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

Fig 3. Control-path diagram

一張含有 文字, 螢幕擷取畫面, Rectangle, 圖表 的圖片

自動產生的描述

Fig 4. Block representation of Control-path diagram

Description:

First, load the firmware code into BRAM and execute it. Then the CPU will execute the program coefficient and data length according to the firmware code. After the program is completed, mprj[23:16] will output a start mark (‘hA5) to notify Testbench to start latency-timer, then the CPU sends X[n] to FIR calculation, and receives Y[n] after calculation. The CPU will then repeatedly send X[n] and receiving Y[n] until data length of Y[n] is

received. when finish, the CPU write final Y[7:0] output to mprj [31:24], and write EndMark(‘h5A) to mprj[23:16], then record the latency timer, Testbench check correctness by checking mprj[31:24], and print out the latency-timer, and finally repeat three times(The CPU output the start Mark(‘hA5) on mprj[23:16], sends x[n] to FIR, receives Y[n] from FIR and until the data length of Y[n] is received and write the final Y[n] to mprj[31:24] and output the EndMark(‘h5A) to mprj[23:16], then check correctness to mprj[31:24]) , and finally record and add up the latency-timer.

**Interface protocol**

一張含有 文字, 螢幕擷取畫面, 平行, 字型 的圖片

自動產生的描述

Fig 5. Interface between firmware and Testbench

一張含有 文字, 螢幕擷取畫面, 行, Rectangle 的圖片

自動產生的描述Fig 6. Interface between firmware and User\_project

**Waveform**

一張含有 文字, 行, 鮮豔, 螢幕擷取畫面 的圖片

自動產生的描述

Fig 7. CPU interaction with Wishbone and mprj\_io

Description:

From the yellow box in Fig 7, you can see that when the CPU sends a request to the user project through wishbone, the wbs\_cyc\_i will be pulled to 1, and then wb\_decode in the user project will determine whether wbs\_cyc\_i is to be given to exmem or FIR, and then the hardware that receives the signal will start executing the behavior. Before executing the behavior, the CPU will

first outputs a start mark(‘hA5) on mprj[23:16] to tell testbench to perform, so you can see the Start mark ('hA5) output by mprj[23:16] from Fig 7, and then the check bit is used by testbench to check whether the mprj output is correct.

**Questions**

What is the FIR engine theoretical throughput?

Description:

It takes 11 cycles to complete a FIR operation, and a total of 64 data are continuously sent to the FIR operation, so it is 11x1+63 = 74 cycles.

Throughput =

What is latency for firmware to feed data?

一張含有 文字, 字型, 螢幕擷取畫面, 資訊 的圖片

自動產生的描述

Fig 8. Show message in display

Description:

Latency = 2373012.500 − 1429912.500 = 943100(ns)

∵ 1ns per cycle, so it is 943100 cycles.

What techniques used to improve the throughput?

* Does bram12 give better performance? In what way?

Description:

Yes, we can put data in one more location to reduce the number of reads and writes, but it will increase the area.

* Can you suggest other method to improve the performance?

Description:

Use circuit techniques such as unfolding, pipelines to increase throughput and performance.

**GitHub**

<https://github.com/ken01235/SOC_Design/tree/master/course-lab_4-2%20report>