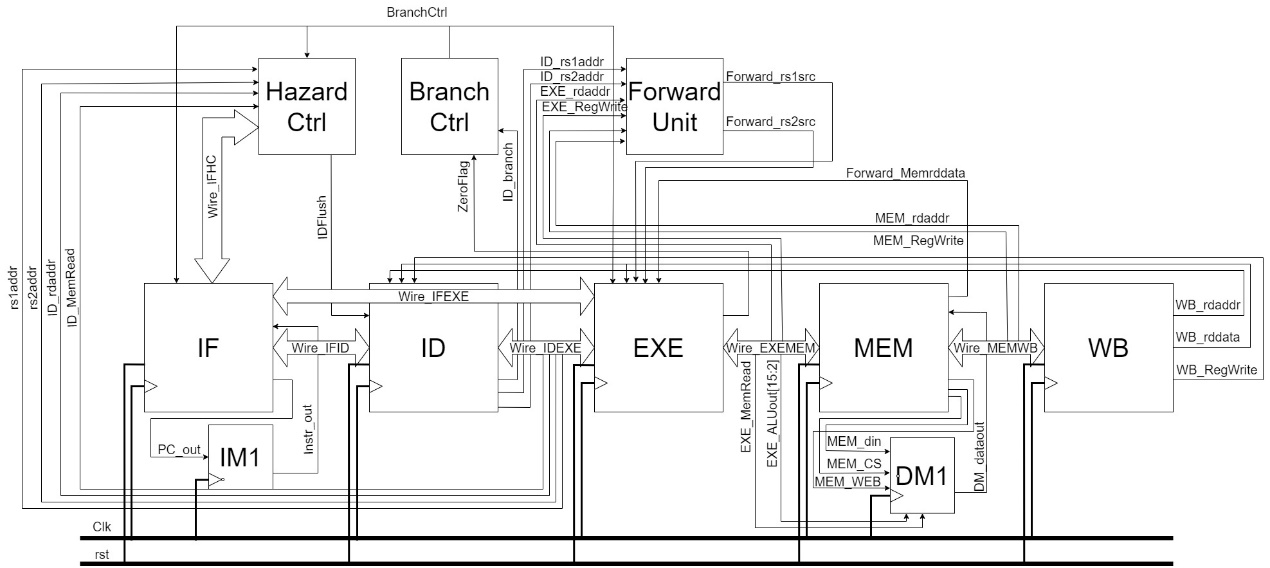
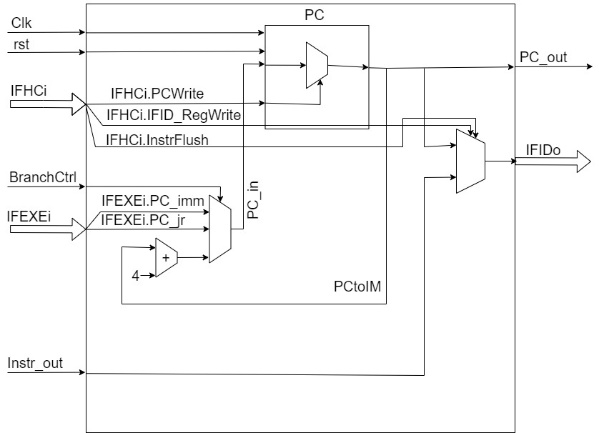
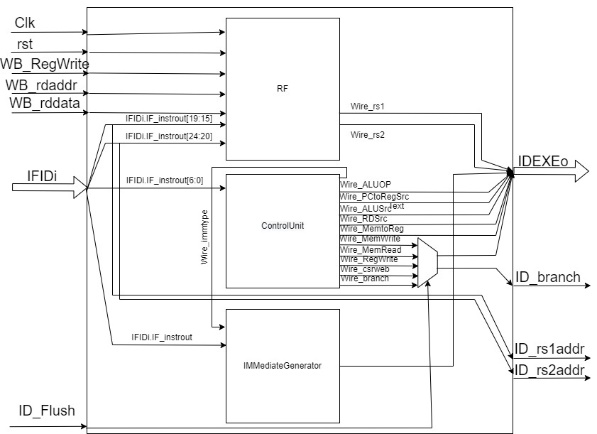
**Explanation & Depict of Design**

1. **Full Design Overview**  
     
   我把CPU分成五個stage，分別是IF、ID、EXE、MEM、WB。且因為目前是使用System Verilog來撰寫程式碼，所以在接線的部分使用了Interface的功能來做嘗試。因為以前在接module時很常因為多接、少接或是更改wire名稱時有缺漏造成debug耗時，因此想藉由此HW學習Interface的使用方式。  
   「正方形」即為此次HW的module，共有10個，其餘module則是接在5個stage之中，在第二部分會仔細說明。  
   「空心雙箭號」即為這次所使用的Interface，內部宣告寫在interface.sv之中，其中在top.sv中實例化作為wire。在各module之間由modport分辨in/output。  
   「實心單箭號」即為單向的資料傳輸，因為只有特定一條或是要輸出給多個module，單獨接線才不會使整體架構太過簡略，造成解讀的困難
2. **Block Detail**  
   **1. IF**



PC主要負責將從PC\_in所得數值傳給PC\_out，由PCWrite來操控PC\_in的值是否可以通過。PC\_in的來源主要有三種，分別是一般、jump以及branch的位置，再由當時BranchCtrl來判斷是不是要跳。

**2. ID**

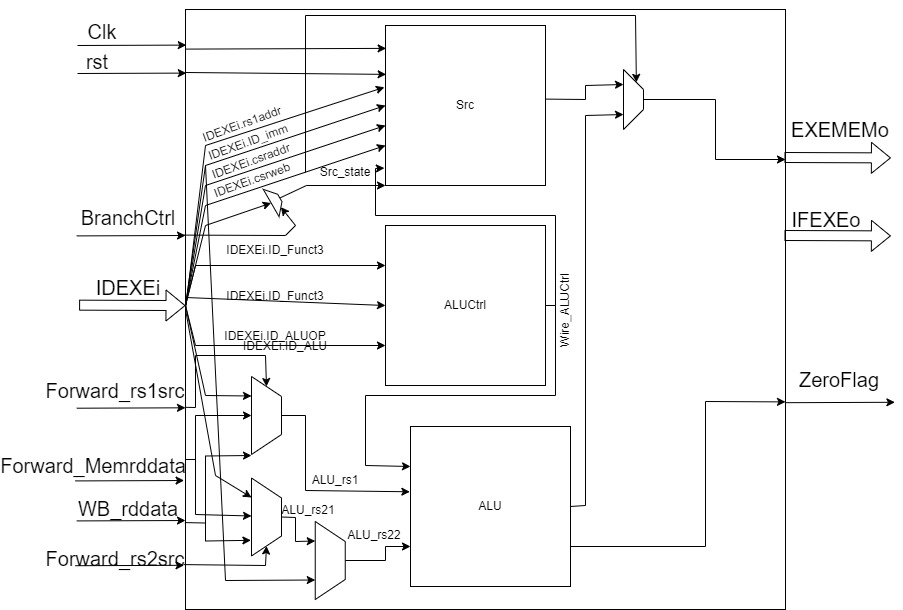


RF用來存值與取值，由data傳到WB時寫回，或是取rs1、rs2的data

CU用來解構instruction，藉由Opcode來決定後續stage的一些function unit signal。且今天如果有Flush時，將相關賦值給予0

IG用來做資料數值擴充，依據CU給的type，來給予不同的擴充方式

**3. EXE**

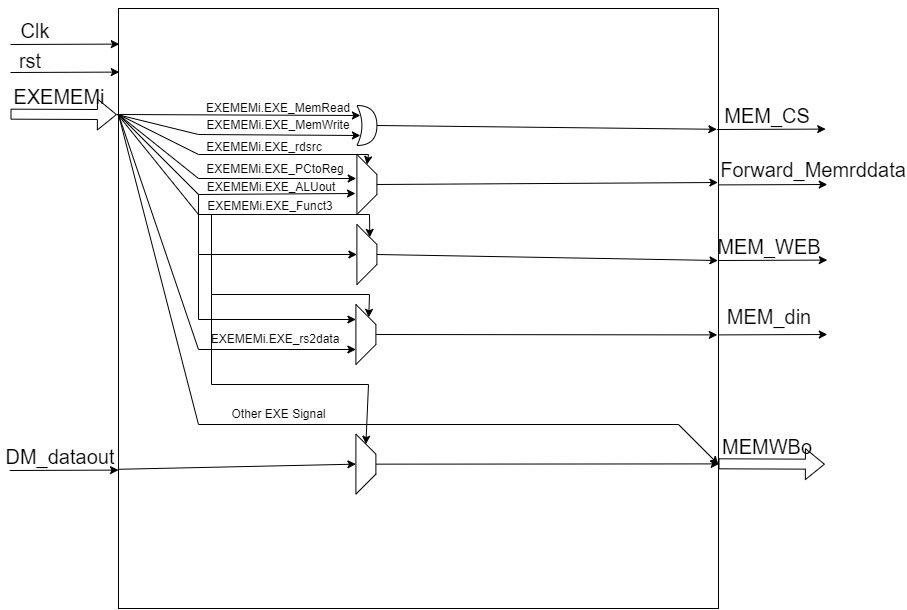


Src主要是計算RDCYCLE以及RDINSTRET，藉由Src\_state來判斷現在的狀態是甚麼，藉此來控制instret的數目。Cycle的部分只要隨著clk做+1即可。而instret又分為branch Flush、Load use以及正常instruction。分別是做-1(因為flush IF&ID)、維持原樣(一加一減不動)以及+1。最後再根據Csrweb來決定是否可以輸出

ALUCtrl 主要是由ALUOP、Funct3以及Funct7來做function的細分，再給ALUCtrl賦值，以利Csr或是ALU做判斷

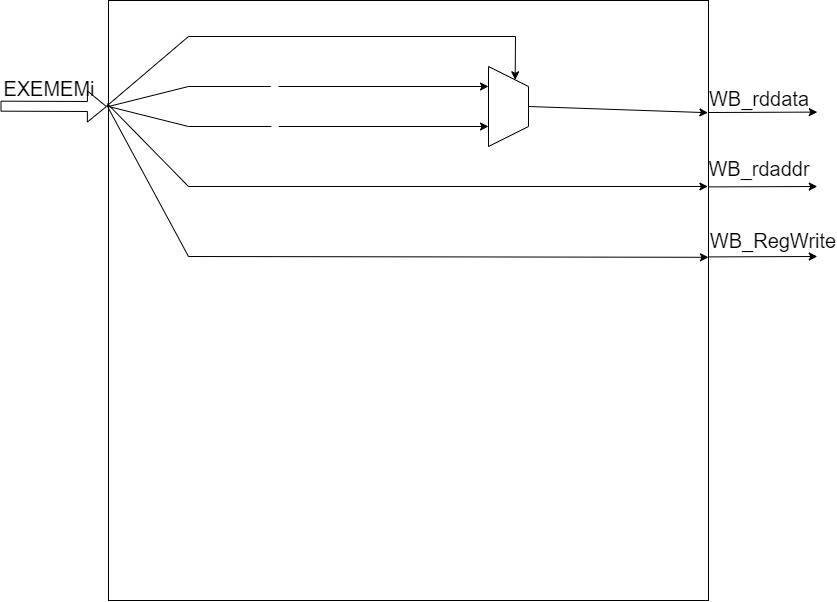
ALU主要是做算術運算(EXE\_ALUout)以及檢查是否有ZeroFlag

**4.MEM**



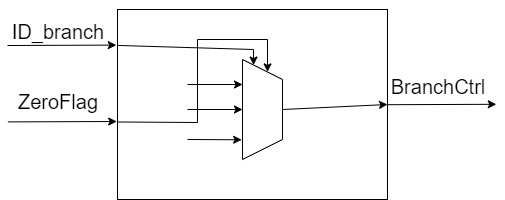
MEM中剩下的主要是一些資料的傳遞以及對DM的lw與sw

**5.WB**



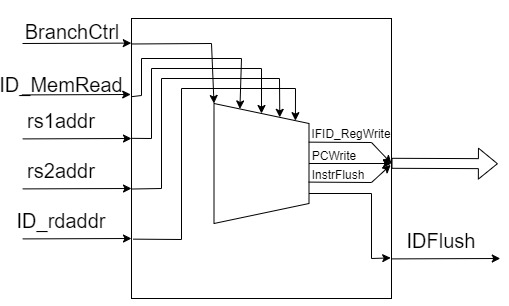
WB也是一樣做資料的傳遞

**6.BC**



BC主要是藉由當下指令狀態，給BranchCtrl做賦值，使得PC中的MUX可以知道下一個指令位置是PC+4或是branch或是jump

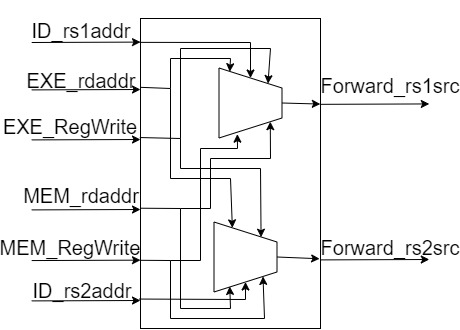
**7.HC**



IFHCo

HC主要是用來看是否需要Flush指令，或是因為有load use，所以要把PC卡住，才可以避免產生錯誤的結果

**8.FU**



FU主要是用來解決一般的data hazard，只要是從EXE算完後馬上要用的值，或是從DM讀出的後也需要做計算時(不包括load use)，在跟ID\_rs1addr以及 ID\_rs2addr做比較，在輸出是要從哪個state中取值(Forward\_rs1,2src)

**三、Lessons learned & Summary** 這次的HW讓我第一次真正的有在做一個project的感覺，因為之前頂多是200~300行可以解決，但是現在做每一個module時都要事先想好每個線路要怎麼佈，然後每個控制訊號是如何互相影響的，另外還要考慮到forwarding、data hazard..等。所以我覺得這次做完這份作業對於RISC-V有更深的了解，雖然還有很多細節是這次沒有的，但我也因為這份作業去翻了RISC-V的規格書，了解到原來之前大學所學不過是簡單中的簡單版本，但是我也收穫良多。其中最有收穫的就屬我會使用interface了，雖然在module的特定output port 要拆線時會比較麻煩，還要另外花時間去網路上學。但是大體上來說節省我很多要接線或是debug的時間，而且學會了以後我覺得對於之後更大型的project時，也更能夠讓其他人一目了然。

**Waveform Verification**

**一、R-type**

1. ADD

2.SLL

3.OR

**二、I-type**

1.LW

2.ADDI

3.JALR

**三、S-type**

1.SW

2.SB

3.SH

**四、B-type**

1.BEQ

2.BLT

3.BGEU

**五、U-type**

1.AUPIC

2.LUI

**六、J-type**

1.JAL

**七、Csr-type**

1.RDINSTRET[H]

2.RDCYCLE[H]

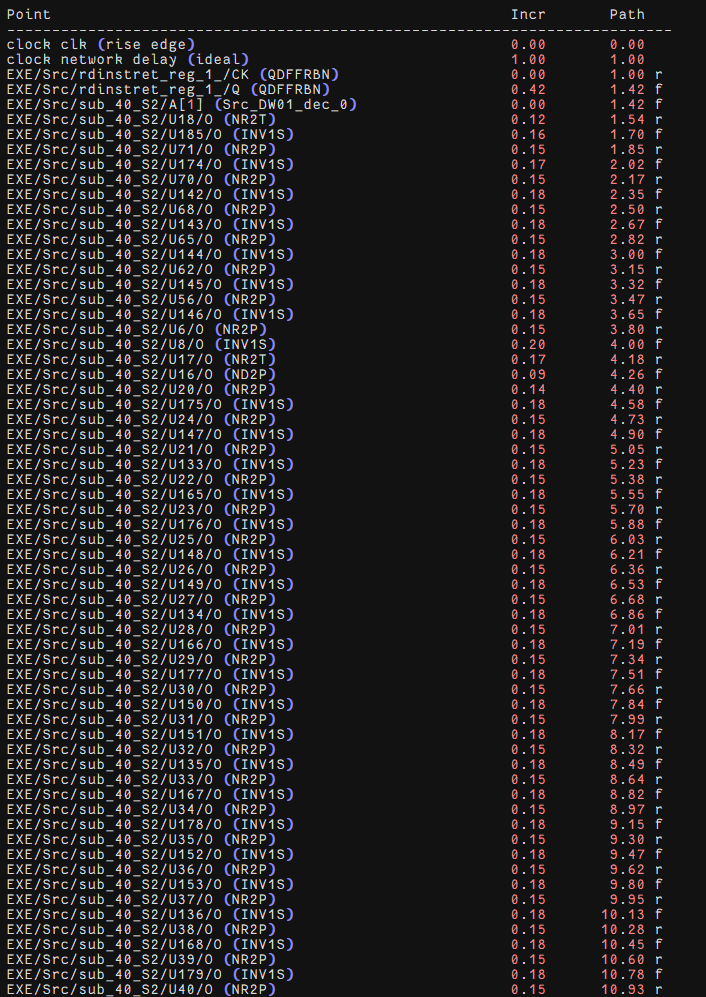
**Report**

**一、Lines of RTL code**

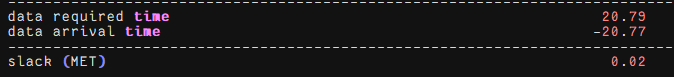
一張含有 文字 的圖片

自動產生的描述

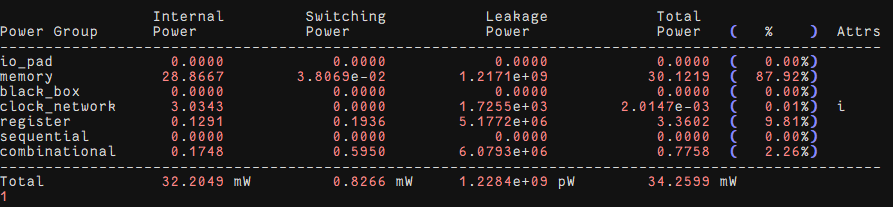
二、**Timing & Area report & Power**

一張含有 文字 的圖片

自動產生的描述



一張含有 文字 的圖片

自動產生的描述

三、**Simulation & Performance**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Prog0 | Prog1 | Prog2 | Prog3 | Prog4 |
| RTL |  |  |  |  |  |
| SYN |  |  |  |  |  |

四、**Result of superlint**

一張含有 文字 的圖片

自動產生的描述

五、**Warning/Errors**

c. Show your snapshots of the waveforms and the simulation results on the terminal for the different test cases in your report and illustrate the correctness of your results.

d. Report the number of lines of your RTL code, the final results of running Superlint and 3~5 most frequent warning/errors in your code. Describe how you modify your code to comply with the Superlint.