DSD Final Project - Pipeline RISC-V Design

Team12
胡芝瑜 連茂翔 朱翔榆

I. Baseline

1. RISC-V pipeline

RISC-V 切成五個 stage, 分別為 IF, ID, EX, ME, WB，以下將由此簡稱進行敘述。此外，為了減少錯誤判斷時需要洗掉的資料數，我們將 branch equal 的判斷移動到 ID stage。

2. I Cache

I cache 連接在 IF stage，為 write back 的 cache。 由於 instructions 連續讀取較多，因此我們使用的是 direct map，且雖然增加到 16 block 會使 miss 大幅減少，但是面積增加過多，AT 值不是表現最好的，因此我們使用的為 8 block，而因為 I cache 並不存在 write 的動作，所以其為 read only。I cache 的 FSM 如 Fig1.

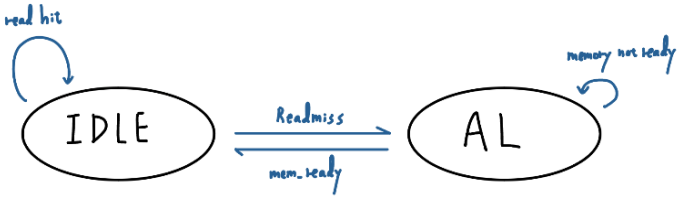

Fig1. FSM of I cahce

3. D cache

D cache 連接在 ME stage, 我們使用的是 2 way associated, 8 block, write back 的 cache。由於我們發現在使用上 read 和 write 要做的事情大多重複，因此我們的 FSM 把 read state 去掉，只留 write state，如 Fig2。

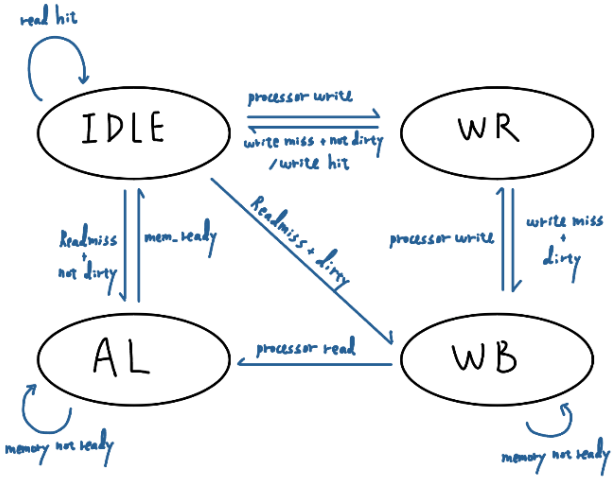

Fig2. FSM of D cache

4. Forward

由於我們的 forwarding 會有從 ME 到 EX，再從 EX 到 ID stage 的路徑，為了避免此 critical path，我們改為只要遇到 EX 到 ID stage 的 forwarding，就會由置入 bobble 避免 hazard 的設計。

II. Compression

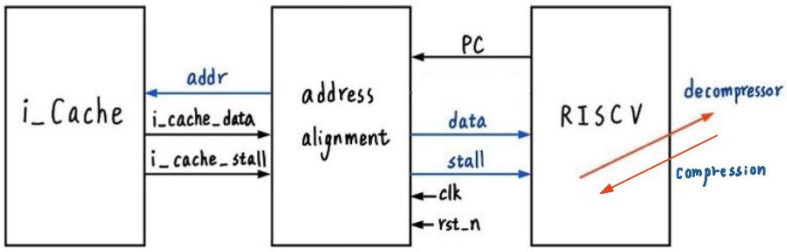

Fig3. Compression in & out

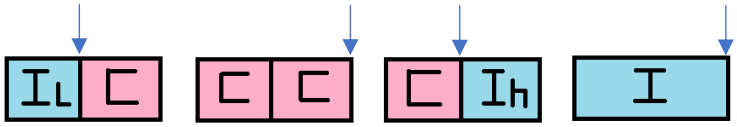

Fig4. Compression conditions

如 Fig3, 我們在 I cache 和 RISC-V 中間插入了 address alignment, 來讀取輸入的 instructions 是否為 compression, 並對應做出不同的輸出改變。

在 Fig4 中可以看到讀取 instruction 可能發生的四種情況，令由左到右為情況 a, b, c, d。

Conditions:

a. 此情況中，由於我們只讀到 32bits instruction 的後半，因此我們會需要 stall 一個 cycle 再等上半部的指令讀取，即為 condition c 右側的 I_h。
   此外，這裡也可能會遇到 instruction 跨 block 的情況發生，此時我們也是採取 stall 一個 cycle 的策略，等進入下一個 block 再進行讀取。

b. 此種情況較為單純，若是再辨別到最後 2bits 代表 16bits 的 compression instruction, 我們則會輸出前面 16bits 全為 0 的 32bits instruction, 並交由 decompressor 辨別 compression 以進行 PC+2，重新組成 32bits 的指令。

c. 此種情況與 b 很相近，只是在 PC+2 的地址中，發現下一個指令為 compression，因此同樣輸出前面為 16bits 0 的 32bits instruction，並由 decompressor 進行後續作業。

d. 此情況則是在最後 2bits 的判別中，確認其為正常的 32bits instruction, 則正常進行輸出與使用。

III. Branch prediction

下列為我們做出來的各種 branch prediction：

1. Always Not Taken, Always Taken
2. 2-bit Saturating Counter
3. 2-level Predictor (Global history, n=2, 3)
   Level predictor 有使用過去的 branch history 選擇要使用哪一個 saturating counter 進行猜測，我們實作的是 n=2, 3 的版本，其中 n=2 為分別由前兩次的 branch 紀錄在 4 個 counter 中選擇，n=3 則是用前三次的紀錄在 8 個 counter 中選擇。
4. 2-level Predictor (Gshare, n=2, 3)

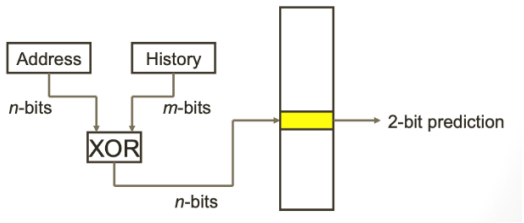

Fig5. Gshare

由 Fig5, gshare predictor 在運作上和 level predictor 很相似，只是改為將 PC 和 history 兩者 XOR 後，再選擇 counter。此作法的好處是可以同時考慮到 global 和 local 的資訊，我們同樣做了 n=2, 3 兩個版本進行測試。

Table1. Branch predictor outcome

	2-bit Saturating Counter	Global Predictor (n=3)	Global Predictor (n=2)
Total cycle (BrPred)	390	375	373
Total cycle (hasHazard)	2431	2310	2304

在 Table1 為我們使用 branch prediction 和 has hazard 進行的測資結果。最右為表現最好的 n=2 的 global predictor，然後才是 n=3 的 global predictor 和 saturating counter。因為表現最好，在針對這兩個測資的部分，我們合成時會使用 n=2 的 global predictor。

IV. Q sort

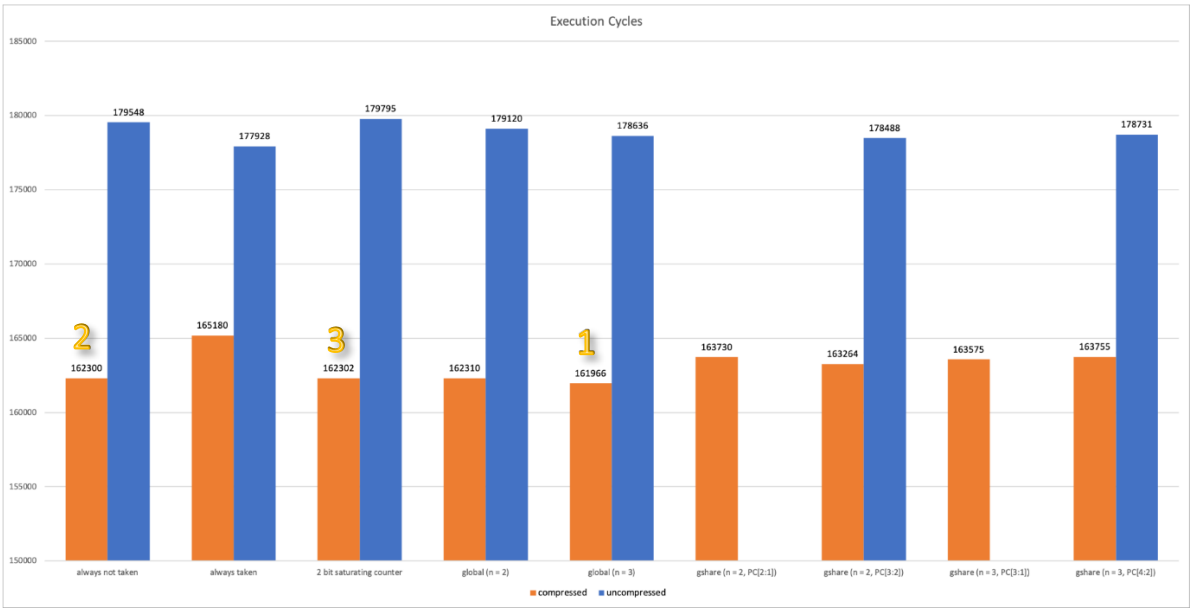

Chart1. Q sort execution cycle

在 Chart1 中，其為所有 branch prediction 針對 Q sort 測試得到的 cycle 數，其中橘色的是 compressed, 藍色的是 uncompressed 的指令。

在 gshare n=2 中，會有兩個是因為 compressed 指令為 PC+2, 即是 PC 最後一位必定是 0，並不能在進行 XOR 時提供足夠的資訊，因此我們有多做選取 PC index 倒數第二和三位數進行 XOR 版本的 gshare(左側的長條圖)；同樣的，因為 uncompressed 指令中為 PC+4，因此我們有多做選取 PC index 倒數第四和三位數進行 XOR 版本的 gshare (右側的長條圖)。

在 gshare n=3 中也相同，其左側為忽略 PC 最後一位，右側為忽略 PC 最後兩位進行 prediction。

在整體而言，compressed 相對於 uncompressed 花得 cycle 數都少許多，因此後續優化也會以 compressed 的版本為主來進行。在這些中，我們針對 cycle 數最少的幾個 predictor，會再進行 cycle time 與 AT 值得比較，如 Table2.

Table 2. Comparison of three branch prediction performance in Q sort AT

	Global Predictor (n=3)	Always Not Taken	2-bit Saturating Counter
Clock cycle (ns)	3.5	2.7	3.5
Execution time (ns)	580592.25	473991.75	581768.25
Area (um^2)	281221.835747	283952.9526	279812.9916
AT score (um^2×ns)	$1.63275x10^{11}$	$1.3459\ x10^{11}$	$1.62786x10^{11}$

Table2 中為 Q sort cycle 數最少的三種 branch prediction, 由左到右增加。我們可以看到雖然 Global predictor 有最少的 cycle 數，但是其在 AT 值上是高於另外兩者。會有此現象的原因，在 2-bit saturating counter 上是因為其面積較小，在 AT 值上些許低於 Global predictor，在 Always not taken 則是因為其 clock cycle 可以壓得較低，因此擁有最好的 AT 值表現。

V. Work distribution

Table3. Work distribution

	朱翔榆	**胡芝瑜**	**連茂翔**
Baseline	Pipeline ALU & ALU control Branch Hazard	Cache Stall Load-use Hazard	Control Forwarding
Extension	Branch Prediction Optimization	Decompressor Compressed PC	Address Alignment Cross Boundary Instruction

VI. Reference

1. https://people.cs.pitt.edu/~childers/CS2410/slides/lect-branch-prediction.pdf
2. Lecture Note