Single Cycle CPU

10801128 陳俊鴻

Q1: 請簡略說明你所新增的檔案內容以及其功能。倘若你沒有新增檔案，則請簡略說明你的Simple_Single_CPU.v 中設計了哪些電路。

A1: 主要參考這次給的 datapath 和 circuit diagram 建立了從 instruction memory 經由 ALU, shifter, 或 zero-filled 做出大部分 arithmetic-logical datapath 和 addi, lui 的功能，再與 data memory 串接整合進 lw 和 sw 的功能，最後再外加 branch 和 jump 必須的電路。

Q2: 根據你設計的電路模組，請解釋在Simple_Single_CPU 中，如何決定arithmetic、logic、shift、load、store、branch 或jump 指令。此外，請列出各個指令在Decoder 的控制訊號輸出值，格式如下表所示。

A2: 透過 decoder 和 ALU control 決定許多 MUX 的選擇和 ALU 的功能。

Main control

	op	RDst	RWrite	M2R	MWrite	MRead	ALUSrc	Branch	BrType	Jump
R	000	1	1	0	0	x	0	0	x	0
addi	001	0	1	0	0	x	1	0	x	0
lui	010	0	1	0	0	x	x	0	x	0
lw	011	0	1	1	0	1	1	0	x	0
sw	100	x	0	x	1	x	1	0	x	0
beq	101	x	0	x	0	x	0	1	0	0
bne	110	x	0	x	0	x	0	1	1	0
jump	111	X	0	x	0	x	X	0	X	1

Q3: 根據你設計的ALU_Ctrl.v 電路模組，列出其輸入與輸出的關係，如下表所示。

A3:

ALU control

	ALUOp[1]	ALUOp[0]	funct	Op[3]	Op[2]	Op[1]	Op[0]	FU[1]	FU[0]
add	1	0	0000	0	0	1	0	0	0
sub	1	0	0001	0	1	1	0	0	0
and	1	0	0010	0	0	0	0	0	0
or	1	0	0011	0	0	0	1	0	0
nor	1	0	0100	1	1	0	0	0	0
slt	1	0	0101	0	1	1	1	0	0
sll	1	0	0110	0	0	1	1	0	1
srl	1	0	0111	1	1	1	1	0	1
addi	0	0	x	0	0	1	0	0	0
lui	1	1	x	x	x	x	x	1	0
lw	0	0	x	0	0	1	0	0	0
sw	0	0	x	0	0	1	0	0	0
beq	0	1	x	0	1	1	0	x	x
bne	0	1	x	0	1	1	0	x	x
jump	x	x	x	x	x	x	x	x	x

Q4: 根據你設計的電路模組，請解釋在Simple_Single_CPU 中，如何決定下一個PC的數值，並且說明各PC 值是從何處計算出來。

A4: 先將PC+2（預設下一個PC），再與 branch address 用 MUX 和 PCSrc 決定是否發生 branch （PC=target address)，最後與 jump address 用 MUX 和 Jump 決定 PC 是否等於 jump address，branch address 和 jump address 之計算與架構圖相符。

Q5: 請寫出你在Lab3 中所碰到的困難，以及實作完成後的心得。

A5: 最大的困難不外乎是不會Verilog和只能用iverilog，因為不會寫test bench所以基本上是一次全部寫完之後試著跑跑看，但因為我不太會Verilog尋找錯誤的過程堪比威利在哪裡，好險stack overflow上也有和我一樣的初學者提出類似問題和好心人士回答問題，但因為有點看不太懂dataflow level和behavior level，所以一開始完全是靠各種gate拼出一個勉強的版本，到後面看多了才運用上一些behavior level的寫法，另外好像Single_Simple_CPU的wire有少一條（決定branch 的 MUX 的output），所以我也自己加上去了。