**Computer Organization 0716074**

* **Architecture diagram:**

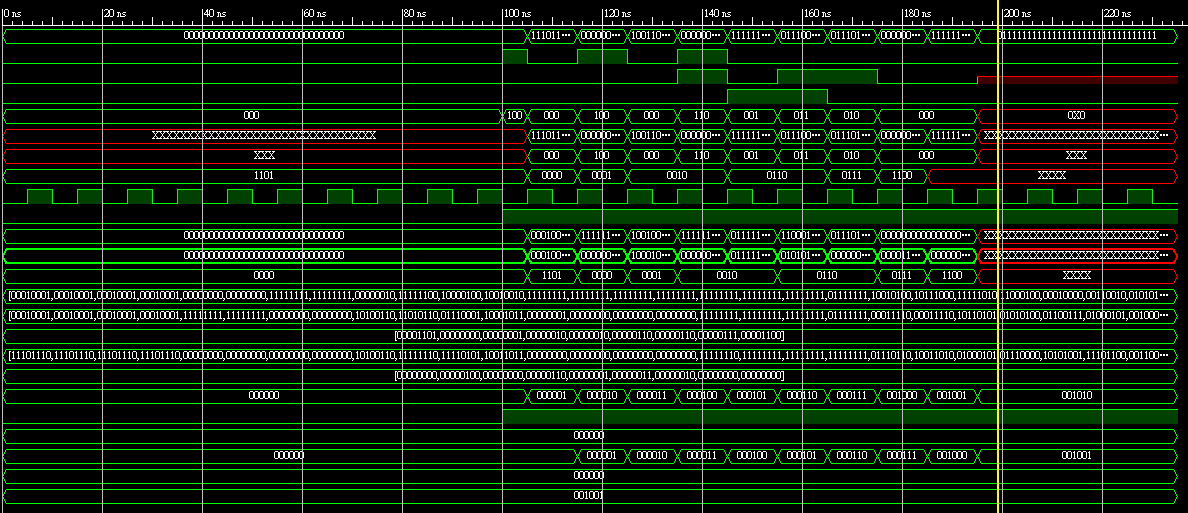
|  |  |
| --- | --- |
| 1. **The structure of 32-bit ALU** | * 本次實作的32位元ALU，有：   Input: **a, b、op code**  Output: **result**  Flag: **Zero, Carryout, Overflow** |
| 1. **alu.v** | * ALU內部的32bits實作上分成兩大部分：  1. 最後一個bit的1-bit ALU (File name is ***alu\_last.v***) 2. 其他的1-bit ALU (File name is ***alu\_1bit.v***) |
| 1. **alu\_last.v** | * 需要實作一個**full adder**   **(full\_adder.v)** |
| 1. **ALU\_1\_bit.v** | * 使用到相同的**full adder (full\_adder.v)** |

* **Detailed description of the implementation:**

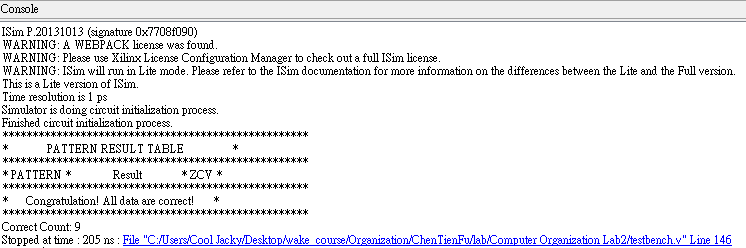
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **Opcode**  |  |  |  | | --- | --- | --- | | **ALU action** | **Function** | **ALU control input** | | and | AND | 0000 | | or | OR | 0001 | | add | Addition | 0010 | | sub | Subtract | 0110 | | slt | Set less than | 0111 | | nor | NOR | 1100 | | nand | NAND | 1101 | | * Opcode分成兩大部分：   + 1. 前2 bits用來決定是否將input a, b做invert:  |  |  |  | | --- | --- | --- | |  | **A\_invert** | **B\_invert** | | **00** | 0 | 0 | | **01** | 0 | 1 | | **10** | 1 | 0 | | **11** | 1 | 1 |  * + 1. 後2 bits用來決定4x1 MUX 哪一個要當result:  |  |  | | --- | --- | | **operation[2-1:0]** | action | | **00** | AND | | **01** | OR | | **10** | Addition | | **11** | Set less than | |
| **B. Detail of *ALU\_1bit.v***  ***module ALU\_1bit(***  ***input src1, //1 bit source 1 (input)***  ***input src2, //1 bit source 2 (input)***  ***input A\_invert, //1 bit A\_invert (input)***  ***input B\_invert, //1 bit B\_invert (input)***  ***input Cin, //1 bit carry in (input)***  ***input [2-1:0] operation, //2 bit operation input set,***  ***output reg result, //1 bit result***  ***output wire cout //1 bit carry out***  ***);*** | * 除了上述的***input A\_invert, B\_invert, operation[2-1:0]***，還加入了addition和subtraction會用到的***input Cin, output cout***，以及輸出該bit結果的***output result***，並且另外加入***set***(預設為0)，給set less than使用，當***opcode = 11***時，***result***都設為***set*** |
| **C. Detail of *ALU\_last.v***  ***module ALU\_last(***  ***input src1, //1 bit source 1 (input)***  ***input src2, //1 bit source 2 (input)***  ***input A\_invert, //1 bit A\_invert (input)***  ***input B\_invert, //1 bit B\_invert (input)***  ***input Cin, //1 bit carry in (input)***  ***input [2-1:0] operation, //2 bit operation***  ***input set,***  ***output reg result, //1 bit result***  ***output wire cout, //1 bit carry out***  ***output wire overflow,***  ***output wire less\_out***  ***);*** | * 最後1bit的ALU多了兩個output: * ***overflow***   最後一個bit的***Cin♁cout***   * ***less\_out***   用在set less than，把最高位加減的結果(也就是signed bit)作為判斷a<b的依據，並將***less\_out***拉到最低位bit的set，即：若a<b，則***less\_out=1***，拉到第一位的***set = 1***，最後此32bits的***result***就會輸出**000…0001** |
| **D. Detail of *alu.v***  ***module alu(***  ***input rst\_n, // negative reset (input)***  ***input [32-1:0] src1, // 32 bits source input [32-1:0] src2, // 32 bits source***  ***input [ 4-1:0] ALU\_control,***  ***// 4 bits ALU control input (input)***  ***output reg [32-1:0] result, // 32 bits result***  ***output reg zero,***  ***// 1 bit when output is 0, zero must be set***  ***output reg cout, // 1 bit carry out***  ***output reg overflow // 1 bit overflow***  ***);*** | * 本次實作的32位元ALU： * Input: ***src1, src2、ALU\_control*** * Output: ***result*** * Flag: ***zero, cout, overflow*** |

* **Implementation results:**

1. **波形圖**

****

1. **結果**

****

* **Problems encountered and solutions:**

**我花了大部分的時間在debug，我唯一錯誤的是第八筆測資，是關於set less than的，不過經過了超久的debug，最後錯誤竟然在要設定B\_invert那裏少加入了ALU\_SLT這個ALU control。**

* **Lesson learnt (if any):**

1. **Divide & Conquer**

讓自己能夠更輕鬆debug的方法就是把每一個物件都排列整齊，利用各種module去實作，而不是直接用基本邏輯式開始硬幹，像是有些selection有時候你把它用mux去做會比每次遇到都用always block更清楚一些。

1. **Learn all of ALU**

經過本次實作，我已經完全了解並且記得所有關於ALU的架構。

1. **Verilog coding**

以前老師只有教紙本的code，現在自己實際寫一次就真的不用硬記了，很自然地就可以看懂verilog並使用。

* **Comment:**

**真心感謝老師和TA精心準備這次的lab，我獲益匪淺，也克服了以前對於機器語言頗難的想像！**