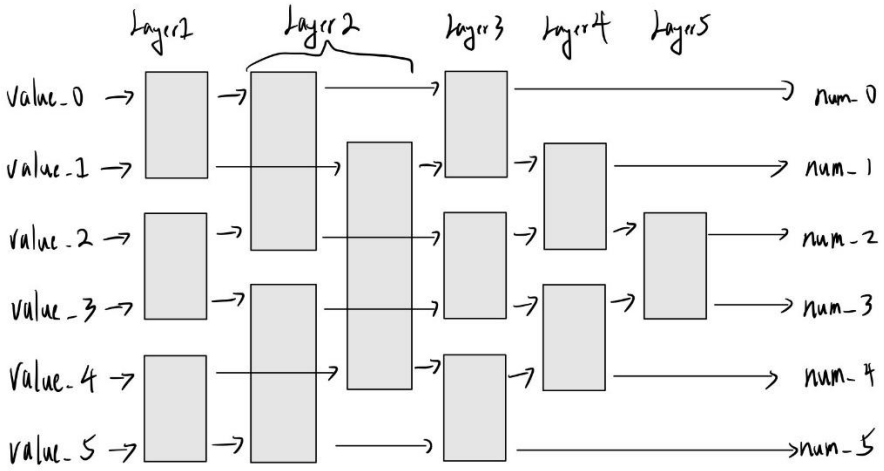
1. 設計方法

首先利用 Connecting Ports by Name list 的方式，將助教寫好的 register 與我的 design 進行連接。

Sorting:

Opcode 有 4 種不同的條件會有不同的排序方式，但仔細看可以更作簡化。當 opcode[4]為 1 時做排序，為 0 時維持原本序列；當 opcode[3]為 1 時將順序變為原本的相反，為 0 時則維持原本序列。

排序使用的是 Merge Sort，使用 12 個比較器完成的，當初設計時先使用 15 個，後來發現若是把距離較遠的 value 先做比較的話可以減少比較器數目。下圖為 Merge Sort 的架構。



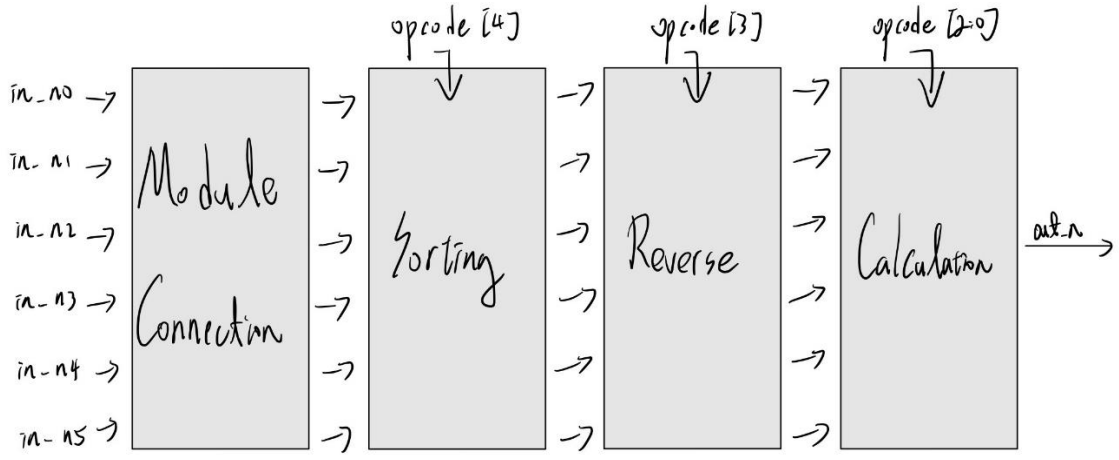
Calculation:

運算的部分利用 case 將題目的條件作為判斷式，再將 out_n 進行題目所要求賦值即可(左圖)。唯獨當 opcode[2:0]=000 時需要再計算平均，以及在此序列中大於此平均的個數為多少(右圖)。

```
//Calculation
case(opcode[2:0])
    3'b000: out_n = count_above_average;
    3'b001: out_n = num0 + num5;
    3'b010: out_n = (num3 * num4) / 2;
    3'b011: out_n = num0 + (num2 << 1); //double
    3'b100: out_n = num1 & num2;
    3'b101: out_n = ~num0;
    3'b110: out_n = num3 ^ num4;
    3'b111: out_n = num1 << 1;
endcase
```

```
if (opcode[2:0] == 000)begin
    sum = num0 + num1 + num2 + num3 + num4 + num5;
    average = sum / 6;
    count = 0;
    if (num0 >= average) count = count + 1;
    if (num1 >= average) count = count + 1;
    if (num2 >= average) count = count + 1;
    if (num3 >= average) count = count + 1;
    if (num4 >= average) count = count + 1;
    if (num5 >= average) count = count + 1;
    count_above_average = count;
end
else begin
    sum =0;
    average=0;
    count=0;
    count_above_average=0;
end
```

2. 架構圖



3. 心得報告

這次的作業還僅僅只是 combinational circuit 而已，因此設計起來並不會是太困難，唯一要注意的還是不能用軟體的思維去寫這份程式，sorting 在 c++中最簡的排序方式只需要兩個 for 迴圈就可以完成，但在撰寫 verilog 中不能使用 for 迴圈，因此會需要花比較多心思在上面。不過，我覺得寫程式都有個共通點就是在寫之前，能夠好好想一次步驟及架構，再用程式去實現，這樣會比較不需要邊寫邊 debug，一次就寫完成的可能性比較大。

這次 Lab3 因為包含了 sequential circuit，對我來說就會比較需要時間思考，希望再多幾次的練習能夠更加熟悉 non-blocking 的寫法。透過每次 lab 及 hw 就能夠將上課所學的知識實際應用，我覺得這種有系統性的學習，未來一定能夠讓我對 verilog 語法更加熟悉。