**DCCDL LAB2**

**matlab**

電機碩一 111521035 林豪澤

使用 rng(32,”twister”); 指令設定亂數產生的種子。

The random sequence we generate with 32 elements is:

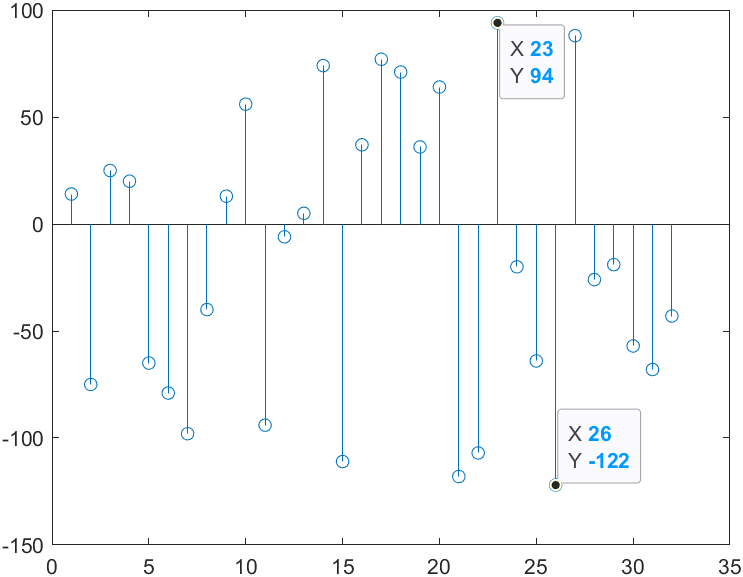
rand\_32 = [14 -75 25 20 -65 -79 -98 -40 13 56 -94 -6 5 74 -111 37

77 71 36 64 -118 -107 94 -20 -64 -122 88 -26 -19 -57 -68 -43]

Y: -128~127

minimum

maximum



𝑥: 1~32

The maximum value with index is 94/23

The minimum value with index is -122/26

將Matrix中的value尾端加上其對應的index，如Matrix中第一個element其值為14，尾數加上其對應的index後即為1401，Matrix第二個element值為-75，經過

處理後為-7502，以此類推。

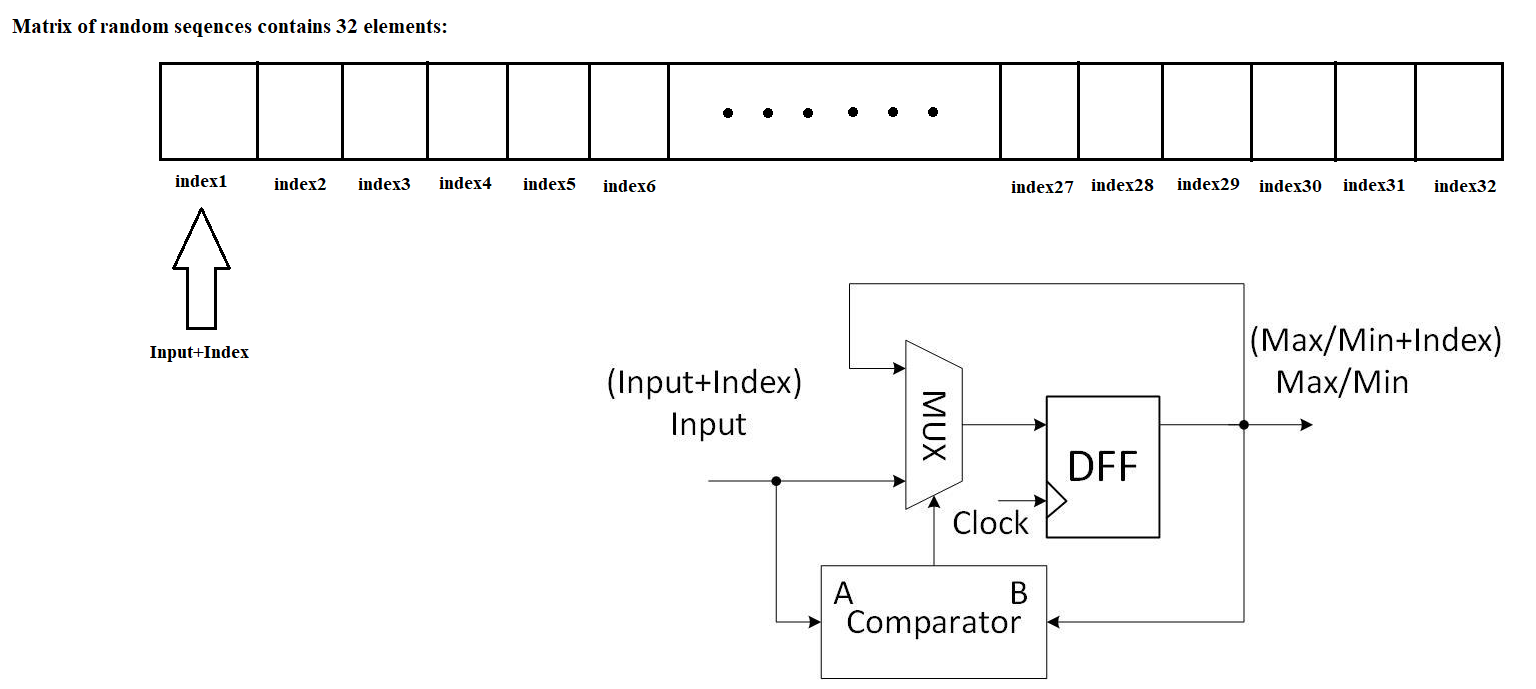
mat\_32\_with\_index = [1401 -7502 2503 2004 -6505 -7906 -9807 -4008 1309

5610 -9411 -612 513 7414 -11115 3716 7717 7118 3619

6420 -11821 -10722 9423 -2024 -6425 -12226 8827 -2628

-1929 -5730 -6831 -4332];

(execution results在背面)

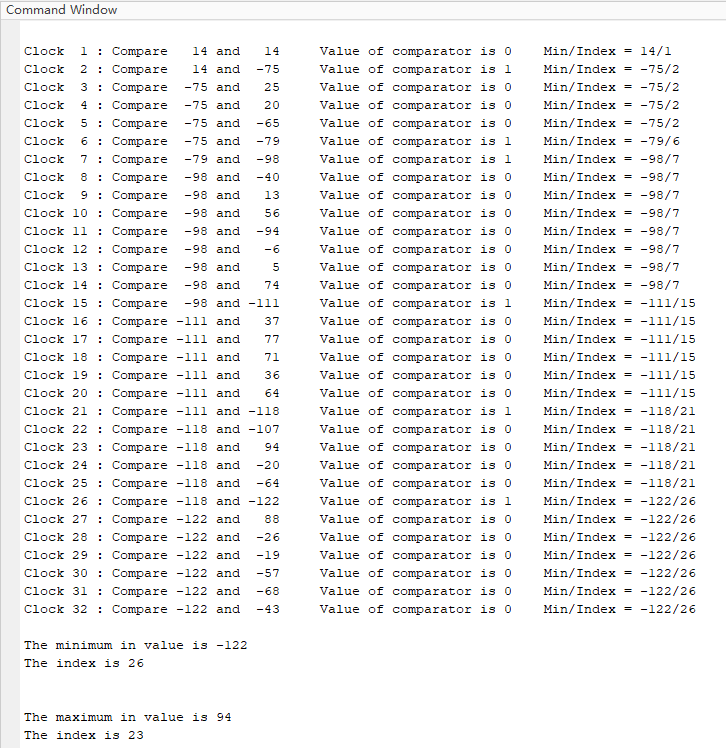


如上圖所示，先將陣列中的第一個element塞入DFF。接著將陣列中的元素逐一使用comparator做比較，找出陣列中的最小值。如果 A<B 則comparator輸出給MUX的訊號即為1，DFF儲存(Input+Index)的值；反之若 A>=B 則 comparator 輸

出給MUX的訊號為0，DFF儲存上一次DFF儲存的值。

Show your execution results including index and value of your sequential comparator.

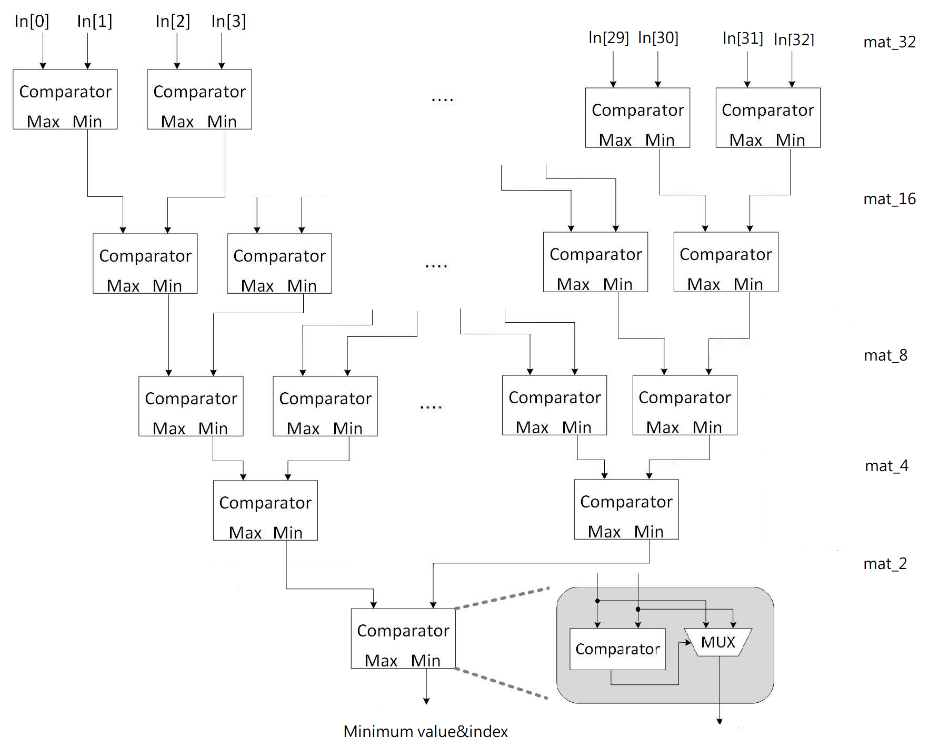
使用32個clock依序比較出陣列中Minimum value與其Index。



1. 由於受到I/O的限制，因此使用32個clock將32個隨機產生的數逐一使用

D-flip-flop讀進mat\_32中。接著如下圖所示，使用parallel comparator將上一層的數值兩兩比較，接者往下傳遞較小值。由此方法花費5個clock找到32個

element中最小的值。



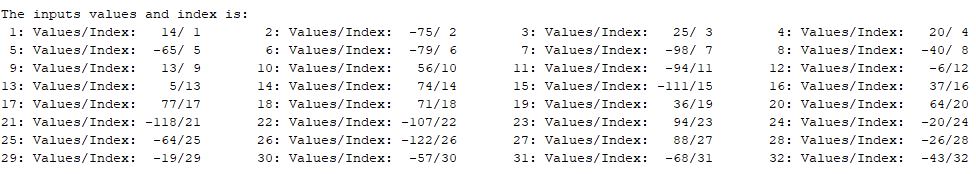
(Use the same inputs as in Q1)

(以下的values皆為加上index的數值，在comparator比較時會以原始值比較)

mat\_32 = [1401 -7502 2503 2004 -6505 -7906 -9807 -4008 1309 5610 -9411

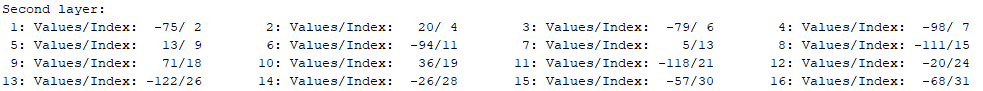
-612 513 7414 -11115 3716 7717 7118 3619 6420 -11821 -10722

9423 -2024 -6425 -12226 8827 -2628 -1929 -5730 -6831 -4332];



mat\_16 = [-7502 2004 -7906 -9807 1309 -9411 513 -11115 7118 3619 -11821 ...

-2024 -12226 -2628 -5730 -6831];



mat\_8 = [-7502 -9807 -9411 -11115 3619 -11821 -12226 -6831];



mat\_4 = [-9807 -11115 -11821 -12226];



mat\_2 = [-11115 -12226];



min\_value =[-12226]



1. Please compare the advantage and disadvantage of serial comparator and parallel comparator.

Serial comparator

Advantage:

所需的I/O腳位較少，comparator也只需要一個就足以完成所有運算工作 -> 需要較少的硬體資源

在元件較少的基礎上，理論上消耗的能源也較少 -> 節省能源

Disadvantage: Serial comparator

採取的是一個時間換空間的戰術，在硬體資源較少的情況下理所當然的，完整處理完同樣資料量的工作也會較parallel長得多。在此條件下

Serial comparator比較適合處理沒那麼具急迫性的資料。Non real-time

Parallel comparator

Advantage:

快速處理資料，serial comparator可能需要32個clock才能處理完的運算，parallel comparator只需要5個clock就能處理完畢。省時快速。

Disadvantage: Serial comparator

採取的是一個空間換時間的戰術，硬體資源相較serial comparator多了一倍不止，不僅如此也需要大量I/O腳位，成本更是多出許多。雖為相較之下的快速解法但是成本與合理性需要經過多方評估才能確定是否為最佳解。

1. Print out your inputs and outputs of your function “Sort4”.

Inputs of function “Sort4” is

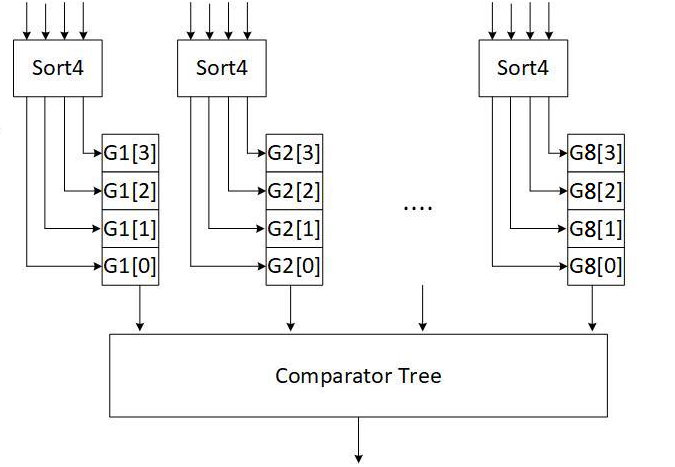
 before\_sort\_4 = [1401 -7502 2503 2004]

Outputs of function “Sort4” is

sort\_4 = [2503 2004 1401 -7502]



1. Print out the 6 outputs of the block SelectTop6 (value and index) among 32 input elements from your program and the results generated by the Matlab command “sort”.



(1)將32個elements 每4個分成一個group，總共有8個group: G1~G8。

以4\*8的陣列將所有32個element儲存起來，每個column為一個group。

(2)先以sort4將每個Group由大至小排列一遍，由此一來每個group中index為 3的位置則為那個Group中的最大值。建立一個index表，標註8個Group中

即將受比較的值。

(3)使用serial comparator 依序比較8個group中每個Group中的最大值，藉此找

到剩下的elements中最大的值。將其儲存在答案的陣列中。

(4)找到此值之後標註其所屬的Group，將其index減掉一，由此一來下次比較時

即不會再次比較到該Group之最大值，而是次大的值。

(5)重複步驟(3)、(4)直到找到32個elements中前六大的值。

以下為

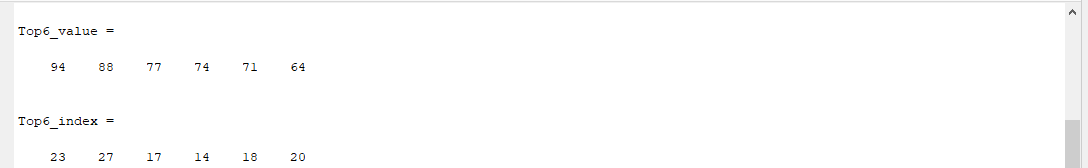
rand\_32 = [14 -75 25 20 -65 -79 -98 -40 13 56 -94 -6 5 74 -111 37

77 71 36 64 -118 -107 94 -20 -64 -122 88 -26 -19 -57 -68 -43];

中使用上述做法所得出的前6大values及其對應的index

Top6\_value = [94 88 77 74 71 64]

Top6\_index = [23 27 17 14 18 20]



以下則是使用”Sort” command產出的答案

