











# Final Project

## Job Assignment Machine

### 1. 問題描述

派工機(Job Assignment Machine) (後文以 JAM表示)的應用相當廣泛，當有n件工作需要完成，而 n 個工人對每件工作可能有不同的工作成本，如何來指派那一個工人去執行那一件工作，以期達到最低成本，此即派工機的目的。

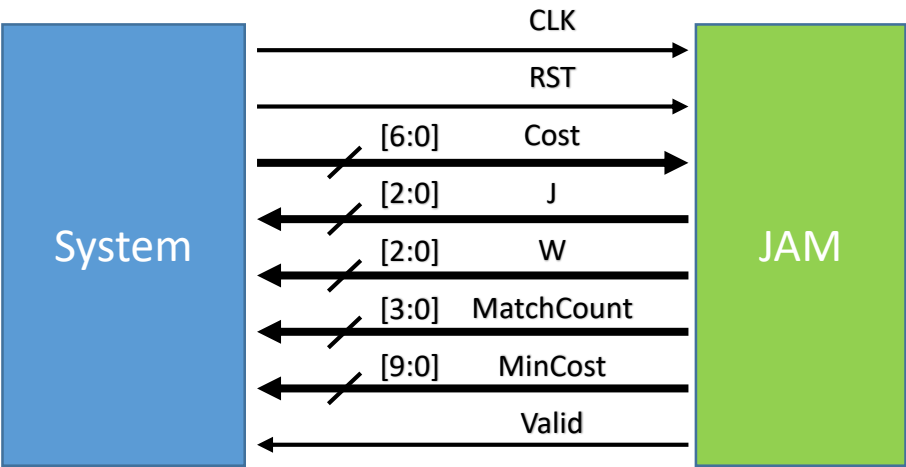
解決派工問題最直覺的方法是計算所有可能組合的成本，然後可找到最低成本的組合。本題題目輸入工人工作成本資訊，請以窮舉法列出所有的配對組合，然後找到最低的成本以及符合最低成本的組合數量。

					
	12	22	34	54	12
	45	21	97	98	34
	54	88	21	22	34
	12	43	57	21	33
	35	98	32	1	13

圖一、工作成本表格

### 2. 設計規格

#### 2.1 系統方塊圖



圖二、系統方塊圖

## 2.2 輸入/輸出介面

表一、輸入/輸出訊號

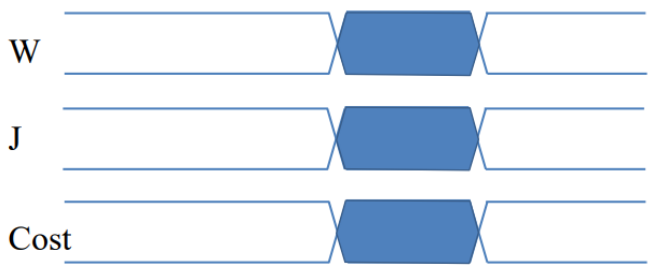
Signal Name	I/O	Wid.	Simple Description
CLK	I	1	Clock Signal (positive edge trigger)
RST	I	1	Reset Signal (active high)。由 testbench 提供，拉高 2 cycle後恢復為 low。
Cost	I	7	成本數值，當 W 及 J 設定完成，Cost 回應第 W 位工人 在第 J 項工作的成本數值。Cost 為無號二進位整數，數值範圍為 0 到 100 的正整數。
W	O	3	指定取得第 W 位工人成本資料, $0 \leq W \leq 7$
J	O	3	指定取得第 J 項成本資料, $0 \leq J \leq 7$
MatchCount	O	4	輸出符合最小成本的可能組合的數量。
MisCost	O	10	輸出最小總工作成本的數值。 MinCost 為無號二進位整數 testbench 的最小總工作成本不會超過 1024。
Valid	O	1	當 Valid 為 high，表示目前 MatchCount 以及 MinCost 為有效的輸出，testbench 會在下一 cycle 立刻結束模擬。

## 2.3 資料輸入與結果輸出描述

題工作成本資料儲存在一塊 8 x 8 的非同步 cost\_rom 中，系統於重置後，JAM 電路指定 W及 J 訊號取得第 W 位工人在第 J 項工作的成本資料。W 及 J 數值範圍皆為 0 到 7。

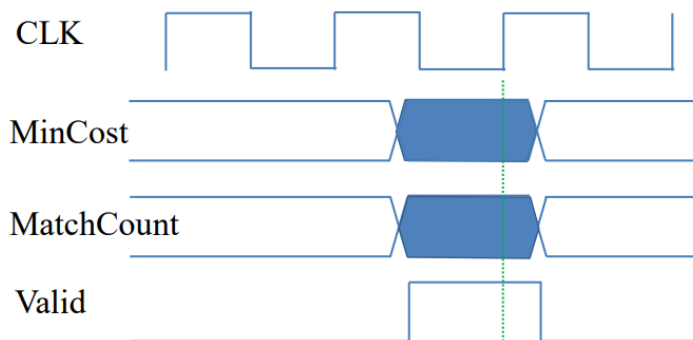
成本資料從 Cost 輸入，直接回應 W 及 J 的內容，不受 CLK 限制。

JAM 電路可重覆取用 cost\_rom 內資料。



圖三、Job Assignment Machine 之輸入訊號波形

JAM 計算出最小總工作成本，以及符合此成本的組合數量後，分別從 MinCost 及 MatchCount輸出，同時拉高 Valid 訊號，testbench 收到 Valid 訊號即會停止模擬並比對結果正確性。



圖四、Job Assignment Machine 之輸入訊號波形

## 2.4 字典序演算法描述

給以窮舉方式產生所有的配對組合，計算符合最小工作成本的組合數量，本題需針對  $n=8$  (指派 8 個工人去完成 8 件工作)的案例進行派工機 JAM 電路作設計。

給定  $n$  個數字，最多可以有  $n!$  種排列，列出這  $n$  個數字所有的排列組合，稱為全排列。以 $n=4$  為例，**[0,1,2,3]**的全排列為：

**[0,1,2,3] [0,1,3,2] [0,2,1,3] [0,2,3,1] [0,3,1,2] [0,3,2,1]**  
**[1,0,2,3] [1,0,3,2] [1,2,0,3] [1,2,3,0] [1,3,0,2] [1,3,2,0]**  
**[2,0,1,3] [2,0,3,1] [2,1,0,3] [2,1,3,0] [2,3,0,1] [2,3,1,0]**  
**[3,0,1,2] [3,0,2,1] [3,1,0,2] [3,1,2,0] [3,2,0,1] [3,2,1,0]**

產生全排列的方式有很多種，在此介紹方法為字典序演算法，字典序的意思是，將序列元素依數字大小排列，比如說**[2,3,1,0]**就排在**[2,3,0,1]**後面，如果有一個方法，可以找到任一序列的下一序列，就可以依續找到所有的序列，這就是字典序演算法。

字典序演算法分成 3 步驟，以下範例以  $n=7$ ，七個數字**[0, 1, 2, 3, 4, 5, 6]**為例，假設目前序列為 **[3, 0, 4, 6, 5, 2, 1]**，求下一字典序列。

1. 從右邊開始，找到第一組相鄰且右邊比左邊大的位置：

以上面序列為例，**[2, 1]** 右邊較小不合條件；

**[5, 2]** 不合條件；

**[6, 5]** 不合條件；

**[4, 6]** 右邊較大，符合條件











我們稱 **4** 的位置為**替換點**，且 **4** 為替換數。**[3, 0, 4, 6, 5, 2, 1]**

2. 在**替換點**右邊的的數字中，找到比替換數大的最小數字，將之和替換數交換  
上面例子，替換數為 4，右邊的數字為[6, 5, 2, 1]，這幾個數中比 4 大的最小數為 5 把 4 和 5 交換，序列變成[3, 0, **5**, 6, 4, 2, 1]
3. 最後把**替換點**後的數字前後順序翻轉過來，即可得下一字典序列。  
把[6, 4, 2, 1]翻轉過來，序列就變成[3, 0, **5**, 1, 2, 4, 6]，此序列就是原序列[3, 0, 4, 6, 5, 2, 1]的下一序列。

## 2.5 計算總工作成本

當計算出某一序列後，即可依此序列數值分配工作給每一位工人，總工作成本計算方式是將每一位工人分配到的工作的成本相加。

比如 n=5 序列[3, 2, 4, 0, 1]，依下圖表格可計算此組合之成本為 54+97+34+12+98

					
	12	22	34	54	12
	45	21	97	98	34
	54	88	21	22	34
	12	43	57	21	33
	35	98	32	1	13

圖五、工作成本表格

### 3. 評分標準

本次作業評分方式會依照設計完成的程度、時間和面積，分為A、B、C、D四種等級。作業**最多兩人**一同協作完成，若三人以上程式碼雷同，則三人**皆視為等級D**，兩人只須要有一人繳交程式碼即可，請在程式碼前面註明兩人的姓名學號。結報(**每人**)內容須包含協作者名稱、程式貢獻比例、程式面積、設計心得、遭遇的困難、解決的方法和學期心得。

#### ◆ 等級A：90～100分

等級A條件：面積最小的100分、第二小99分依此類推，第十名及以後皆為90分  
a、總模擬cycle數小於430,000cycle。

#### ◆ 等級B：80分

等級B條件：  
a、總模擬cycle數小於600,000cycle。

#### ◆ 等級C：59分

等級C條件：  
a、準時繳交學期心得。  
b、總模擬cycle數大於600,000cycle(**再加11分**)。

#### ◆ 等級D：0分

等級D條件：符合下列任一項  
a、未繳交程式。  
b、繳交程式碼功能不齊全。  
c、未通過防抄襲比對。

### 3.1 程式面積

將完成的程式，使用 Quartus 軟體合成，並記錄下程式使用的邏輯閘和暫存器數量。

Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	764 / 49,760 (2 %)
Total registers	289
Total pins	36 / 360 (10 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	2 / 288 (< 1 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

圖七、Quartus合成的數據報告

## 4. 附錄

附錄4.1~3為設計檔案說明；附錄4.4為測試樣本；

### 4.1 設計檔案說明

表二、設計檔案說明

檔名	檔案說明
JAM.v	所有學生使用的設計檔，包含輸入輸出宣告。
tb.v	TestBench檔案。
cost_rom	測資資料。

### 4.2 測試檔內容

本次練習所提供的testbench檔案和設計檔，有多增加幾行特別用途的敘述：

```
“tb.v”
`define PAT "cost_rom"
`define PAT_NUM      2
`define End_CYCLE    1000000

-----
“JAM.v”
module JAM(.....);
...
for(i = 0; i < `length; i = i+1)
    $dumpvars(1, count[i]);
...

```

(可以自行更換測資1~3)  
(可以自行決定最終運行時間)

(若要在gtkwave輸出多維陣列時，需加此行)

### 4.3 特殊情況

若在執行的過程中出現以下警告，可以忽略：

```
VCD info: dumpfile Lab.fsdb opened for output.
VCD warning: array word Pin_Name will conflict with
an escaped identifier.
```

## 4.4 測試樣本

Pattern1 :

MinCost=119 , MatchCount=3

Min cost at job serial : (in order from W0 to W7)

1 7 5 0 4 3 6 2

6 7 5 0 4 3 1 2

6 7 5 0 4 3 2 1

	J0	J1	J2	J3	J4	J5	J6	J7
W0	11	25	53	41	59	32	25	59
W1	4	11	25	11	59	31	53	11
W2	11	59	15	11	15	15	53	53
W3	4	59	32	34	53	41	34	59
W4	15	32	41	34	4	59	34	32
W5	41	59	59	4	4	41	34	34
W6	53	31	25	41	59	32	31	53
W7	11	31	25	11	34	34	53	32

Pattern2 :

MinCost=250, MatchCount=6

Min cost at job serial : (in order from W0 to W7)

4 1 2 0 3 5 6 7

4 1 2 0 3 6 5 7

4 1 2 0 7 5 6 3

4 1 2 0 7 6 5 3

4 1 2 3 7 5 6 0

4 1 2 3 7 6 5 0

	J0	J1	J2	J3	J4	J5	J6	J7
W0	54	59	59	59	32	40	62	40
W1	54	32	32	79	32	38	32	62
W2	54	54	30	38	32	38	59	54
W3	30	59	32	32	62	40	45	79
W4	32	32	38	32	62	38	62	32
W5	79	45	32	62	32	32	32	59
W6	32	38	32	59	54	30	30	45
W7	30	79	32	32	62	30	45	32

Pattern3 :

MinCost=485, MatchCount=9

Min cost at job serial : (in order from W0 to W7)

1 5 2 7 4 0 3 6

1 5 4 7 6 0 3 2

2 5 4 7 6 0 3 1

3 5 2 7 4 0 1 6

3 5 4 7 6 0 1 2

5 1 2 7 4 0 3 6

5 1 4 7 6 0 3 2

5 4 2 7 6 0 3 1

5 6 2 7 4 0 3 1

	J0	J1	J2	J3	J4	J5	J6	J7
W0	81	60	60	65	96	60	65	96
W1	96	60	66	96	60	60	60	81
W2	96	66	60	99	60	81	65	65
W3	66	96	80	99	81	81	96	60
W4	81	96	65	96	60	96	60	81
W5	60	96	80	96	80	60	81	60
W6	99	60	99	65	80	80	81	66
W7	65	60	60	99	99	80	60	96