NYCU-ECE DCS-2022

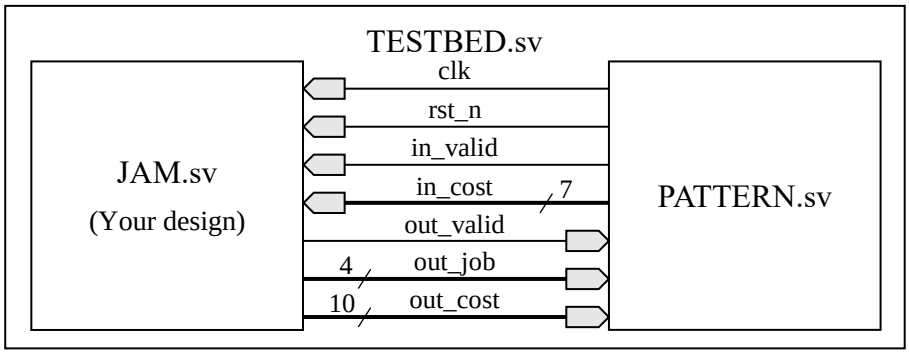
Final Project

Design: Job Assignment Machine

資料準備

1. 從 TA 目錄資料夾解壓縮:
   % tar -xvf ~dcsta01/Final.tar
2. 解壓縮資料夾 Final 包含以下:
   a. 00_TESTBED/
   b. 01_RTL/
   c. 02_SYN/
   d. 03_GATE/
   e. 09_UPLOAD/

Block Diagram



設計描述

派工機(Job Assignment Machine, 以下簡稱JAM)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。本次Final Project要針對 **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。

1. JAM電路資料輸入:

當in_valid為high時，工作成本資料由in_cost訊號輸入，工作成本資料輸入的順序先以第1位工人的第1項工作~第8項工作成本連續輸入8個cycle後，接著輸入第2位工人的8個工作成本連續8個cycle，以此類推，如下圖所示，總共連續輸入64 cycle的工作成本。

Cost	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8
Worker 1								
Worker 2								
Worker 3								
Worker 4								
Worker 5								
Worker 6								
Worker 7								
Worker 8								

2. JAM電路輸出:

接收到8位工人對應8項工作各別的工作成本後，要找出所有組合中最低成本的組合並且計算出最低成本。

以下圖為例，當第1至第8位工人分別對應到的工作編號為**[2, 3, 6, 8, 1, 5, 4, 7]**，可以得到最低成本 20 + 29 + 23 + 23 + 21 + 26 + 21 + 36 = 199。

輸出資料時out_valid要維持high連續8個cycle，out_job訊號在這8個cycle內要依序輸出第1至第8位工人分別對應到的工作編號，如上圖例子即為2、3、6、8、1、5、4、7，而out_cost訊號要連續8個cycle都是輸出最低成本199。

Cost	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8
Worker 1	49	**20**	32	38	35	50	50	44
Worker 2	40	25	**29**	42	37	33	43	31
Worker 3	28	44	20	29	34	**23**	44	33
Worker 4	38	20	29	31	47	26	37	**23**
Worker 5	**21**	48	50	28	40	29	40	46
Worker 6	21	39	29	38	**26**	29	42	41
Worker 7	47	20	39	**21**	31	29	23	39
Worker 8	49	27	39	43	35	32	**36**	27

當遇到可形成最低成本的組合有兩種以上的情形時，則以編號越小的工人得到編號越小的工作的工作分配組合為優先。

如下圖範例，當第1至第8位工人分別對應到的工作編號為**[3, 6, 8, 4, 5, 1, 2, 7]** (紅色與橘色標示之數字)與**[3, 8, 6, 4, 5, 1, 2, 7]** (紅色與綠色標示之數字)兩種組合時，都可以得到最低成本182。

從第1位工人開始看：第1位工人在兩種組合中相同，都會分配到第2項工作，接

著看第2位工人；第2位工人在第一種組合分配到第6項工作，而在第二種組合則會分配到第8項工作，第6項工作編號較第8項小，因此下圖表格範例要以第一種組合作為答案，out_job依序輸出3、6、8、4、5、1、2、7，out_cost輸出182連續8個cycle。

Cost	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8
Worker 1	28	66	**26**	68	66	42	64	76
Worker 2	24	18	76	42	46	**28**	74	**14**
Worker 3	62	40	58	68	14	**34**	68	**20**
Worker 4	28	68	44	**24**	40	72	64	50
Worker 5	50	48	74	46	**24**	50	52	68
Worker 6	**22**	60	70	34	58	32	18	28
Worker 7	32	**24**	32	26	34	40	50	50
Worker 8	40	40	32	76	34	42	**14**	62

3. JAM電路演算法

本次Final Project不限定使用的演算法，Job Assignment Problem可以在網路上找到許多相關資料與演算法，常見的演算法有**暴力破解法(窮舉法)、匈牙利演算法、深度優先搜尋(DFS)、廣度優先搜尋(BFS)演算法**，同學可以依照自己所採用的演算法去設計電路，只要最後輸出符合上述規定即可。

4. Pattern

本次Final Project的Pattern、測資都要由你們自行完成去測試你們自己的電路，Pattern要檢查的地方請參照下方 Specifications 2~8，測資規則請參照上方說明。

助教提供5組的範例測資在00_TESTBED中的input.txt和output.txt，input.txt每8行為一組測資，資料順序與上方說明相同，output.txt每1行為一組測資，前8個數字為out_job的順序，第9個數字為out_cost。

input.txt

job i

worker i

```
1  70  119 75  55  72  107 18  97
2  57  81  101 9   97  97  126 102
3  26  88  101 15  65  85  111 125
4  7   12  22  111 39  123 48  34
5  119 127 61  117 43  24  16  14
6  111 22  74  18  41  66  54  106
7  75  45  58  38  5   115 86  32
8  86  81  113 122 104 98  87  109
9  75  9   109 78  115 109 110 110
10 29  91  53  43  54  65  47  8
11 9   97  56  123 59  126 37  8
12 49  117 31  37  79  17  47  14
13 96  66  53  31  10  106 6   46
14 59  53  102 19  81  51  48  43
15 123 69  36  117 105 39  92  118
16 106 16  33  75  103 91  73  47
⋮
```

output.txt

```
1  7 4 1 3 8 2 5 6 214
2  2 8 1 7 5 4 6 3 174
3  6 5 2 3 8 1 4 7 94
4  7 8 5 4 2 3 1 6 165
5  4 3 1 8 7 5 2 6 179
```

out_job　　out_cost

Inputs

Signal name	Bit width	Description
clk	1	200MHz clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	Getting high when **in_cost** is valid.
in_cost	7	Cost data, transmitted in 64 cycles continuously.

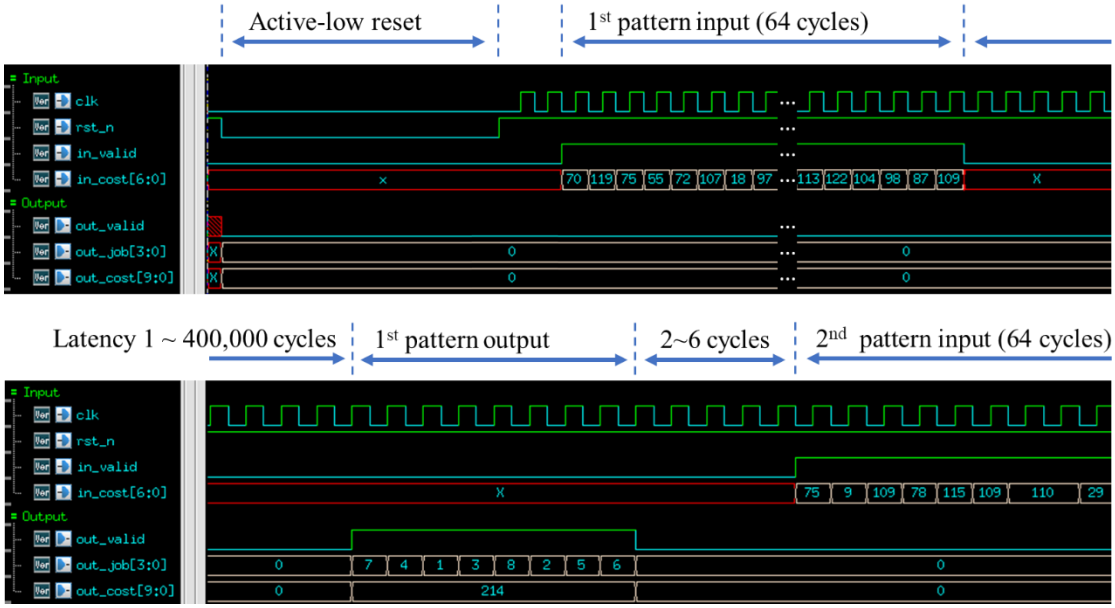
Outputs

Signal name	Bit width	Description
out_valid	1	Getting high when **out_job** and **out_cost** are valid.
out_job	4	Job assignment index, transmitted in 8 cycles continuously. The data at n-th cycle presents the job index the n-th worker will get.
out_cost	10	The lowest cost summation, transmitted in 8 cycles continuously.

Specifications

1. Top module name: **JAM** (File name : **JAM.sv**)
2. 只會在模擬一開始 reset 一次，
   在非同步負準位 reset 後，所有的 output 訊號必須歸零。
3. Input 訊號連續輸入 64 cycles。
4. Output 訊號要在 Input 結束後的 400,000 cycles 內輸出。
5. **out_valid** 訊號不能與 **in_valid** 有任何重疊。
6. 每筆測資 Output 訊號連續輸出 8 cycles，所有答案必須完全正確。
7. 所有 Output 訊號都要在輸出結束後歸零。
8. 下一筆測資的 Input 會在上一筆的 Output 結束後的 2~6 cycles 後開始給值。
9. Clock period 5 ns.
10. Input delay = 0.5 * clock period; Output delay = 0.5 * clock period.
11. 02_SYN result 不行有 error、不能有任何 latch、不可以 timing violation。
12. 03_GATE 不能有任何 timing violation。
13. 03_GATE 的 Latency 要與 01_RTL 相同。
14. Separate your combination and sequential logic.

Example waveform



上傳檔案

1. Upload your **Code** through the script in 09_UPLOAD folder.
2. Upload your **Report** to NewE3 platform.
   Naming Rule: **report_dcsxx.pdf**, xx is your server account.
3. Deadline:
   Demo1: **6/10 23:59 pm** & release TA demo pattern.
   Demo2: **6/17 23:59 pm**.
   Report: **6/17 23:59 pm**.

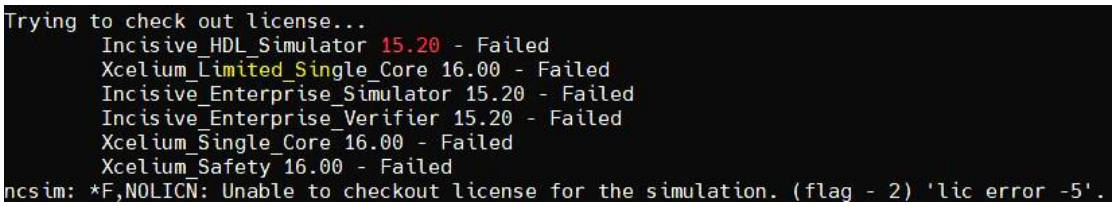
Grading policy

1. Pass the RTL simulation & Synthesis. **(30%)**
2. Pass the Gate-level simulation. **(30%)**
3. Performance = Area * Latency **(30%)**
4. Report **(10%)**
5. Coding Style (If you do not follow the Spec. 14) **(-5%)**

Note

Template folders and reference commands:

1. 01_RTL/ (RTL simulation) → **./01_run**.
2. 02_SYN/ (synthesis) → **./01_run_dc**
3. 03_GATE/ (gate-level simulation) → **./01_run**.
4. 09_UPLOAD/ (upload) → **./01_upload**



If the license failure occurs while running 01_RTL or 03_GATE,
try one of these two commands:

→ **source ~dcsta01/license_1.cshrc**

→ **source ~dcsta01/license_2.cshrc**

報告請簡單且重點撰寫，不超過兩頁A4，並包括以下內容

1. 描述你的設計方法，包含但不限於如何加速(減少critical path)或降低面積。
2. 基於以上，畫出你的架構圖(Block diagram)
3. 遇到的困難與如何解決。
4. 心得報告，不侷限於此次作業，對於作業或上課內容都可以寫下。