2022 IC Design Contest

Cell-Based IC Design Category for Graduate Level

Job Assignment Machine

1. Problem Description

Job Assignment Machine (**JAM**) has a wide range of applications and the purpose of JAM lies in the following: Assuming there are n jobs need to be done and n workers contain different job costs for each job, how to assign which worker for which job so that the cost is minimized.

The most intuitive approach for solving the JAM problem is to calculate costs for each possible combination and identify the one that delivers the lowest. In this problem, the information about the job cost for workers is given, please list all the possible combinations, find the lowest cost, and the number of combinations that contains the lowest cost.

The detailed specifications about the JAM will be illustrated later. Table 1 lists inputs/outputs signals with functionality descriptions for this circuit. Each team needs to finish the design and verification according to the design specifications that will be presented in the next section.

This IC design contest is from 8:30am to 8:30pm. When the contest is finished, the grading will be conducted according to the standard that is depicted in section three. In order to facilitate the grading process, each team should submit the files that are required for grading listed in appendix C.

	⛏	🪣	🍴	⁘	🕺
👨‍🍳	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

Fig. 1、Table for the Job Costs

2. Design Specifications

2.1 System Block Diagram

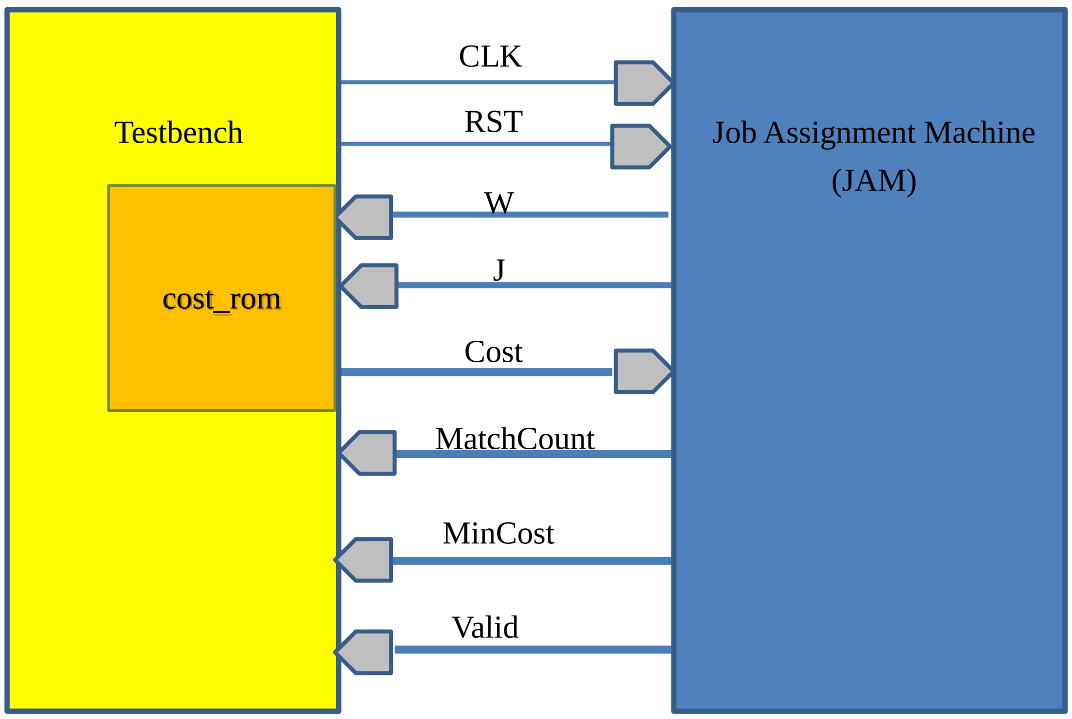


Fig. 2、Job Assignment Machine System Block Diagram

2.2 Inputs/Outputs Interfaces

Table 1、Inputs/Outputs Signals

Signal Name	I/O	Width	Simple Description
CLK	I	1	Clock Signal (positive edge trigger)
RST	I	1	Reset Signal (active high)。Provided by testbench, restoring to low 2 cycles after pull-up.
W	O	3	Assign to acquire the cost information for the W-th worker, $\mathbf{0 \leq W \leq 7}$
J	O	3	Assign to acquire the the J-th cost information, $\mathbf{0 \leq J \leq 7}$
Cost	I	7	The value of the cost. When W and J are assigned, Cost responds with the value of the cost for the W-th worker on the J-th job. Cost is an unsigned integer with the range between 0 and 100.
MatchCount	O	4	Output the number of combinations with minimum cost.
MinCost	O	10	Output the value of the minimum total job cost. MinCost is an unsigned integer. The minimum total job cost is not larger than 1024 in testbench.
Valid	O	1	When Valid is high, the MatchCount and MinCost are valid output, and the testbench finishes the simulation in the next cycle.

2.3 System Description

List all possible combinations and calculate the combination that delivers the minimum job cost. In this problem you need to design the circuit of JAM for the case of n = 8 (assigning 8 workers for 8 jobs).

2.3.1 Input of the original cost data to JAM circuit

The job cost data is stored in a 8 x 8 synchronized cost_rom. After reset, the JAM circuits assigns W and J signals to acquire the cost data for the W-th worker on J-th job. The range of W and J are both from 0 to 7. The cost data is input from Cost, and responds to W and J on the rising edge of CLK.

The JAM circuit can access the cost_rom repeatedly.

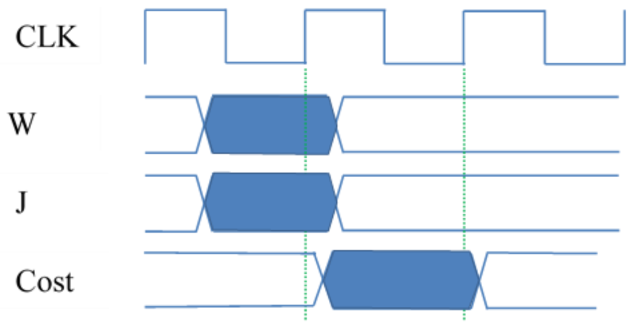


Fig. 3、The Input Signals for the Job Assignment Machine

2.3.2 Output for the JAM result

After calculating the minimum job cost and the number of combinations for this minimum job cost, the results are output through MinCost and MatchCount. Meanwhile, the Valid signal is pull-up. The testbench will stop simulation and compare the results once the Valid signal is received.

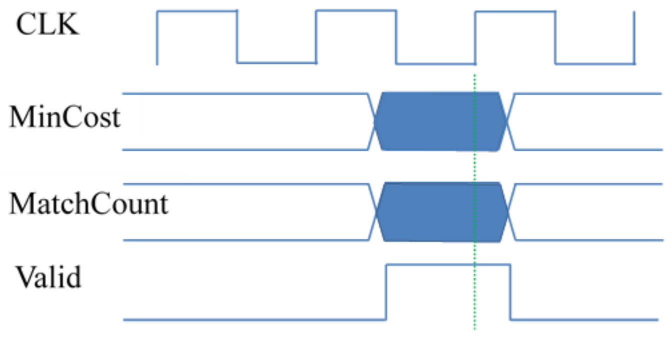


Fig. 4、The Output Signals for the Job Assignment Machine

2.3.3 The Full Permutation Algorithm

Given n numbers, there are as many as n! permutations. The full permutation algorithm is to list all possible permutations based on the n numbers. For an n = 4 example, the full permutation of **[0,1,2,3]** is:

[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]**

Here introduces the lexicographical algorithm where the elements are ordered according to the number. For example, **[2,3,1,0]** is ordered after **[2,3,0,1]**. The lexicographical algorithm is to find the next sequence of any sequence and then so on and so forth…

The lexicographical algorithm contains three steps and following illustrates an example of n = 7 with **[0, 1, 2, 3, 4, 5, 6]**. Assuming the current sequence is **[3, 0, 4, 6, 5, 2, 1]**, find the next sequence:

1. Starting from the right, find the first neighboring location where the right-hand side (RHS) is larger than the left-hand side (LHS):

   Take the aforementioned sequence as the example, **[2, 1]** does not meet the condition since RHS is smaller than LHS ; **[5, 2]** does not meet the condition ; **[6, 5]** does not meet the condition. **[4, 6]** meet the condition since RHS is larger than LHS. We call the position of **4** the replacement point and **4** is the replacement number. **[3, 0, $\boxed{4}$, 6, 5, 2, 1]**
2. For the numbers on the RHS of the replacement point, find the minimum number that is larger than the replacement number and exchange that number with the replacement number. For this example, the replacement number is **4** and the numbers on the RHS are **[6, 5, 2, 1]**. The minimum number that is larger than **4** in this sequence is **5**. Thus **4** and **5** are exchange and the sequence becomes **[3, 0, $\boxed{5}$, 6, 4, 2, 1]**.
3. Finally, the order for the numbers after the replacement point is flipped to get the next sequence. For example, to flip the order of **[6, 4, 2, 1]** to get the sequence of **[3, 0, $\boxed{5}$, 1, 2, 4, 6]**. This sequence is the next sequence of **[3, 0, 4, 6, 5, 2, 1]**.

2.3.4 To Calculate the Total Job Cost

After the calculation of a sequence, the job can be assigned to each worker according to this sequence. The total job cost is calculated as adding the cost of every job that is assigned to each worker. For an example of n=5 with sequence of [3, 2, 4, 0, 1], the total job cost can be calculated as 54+97+34+12+98 according to the Table shown as follows.

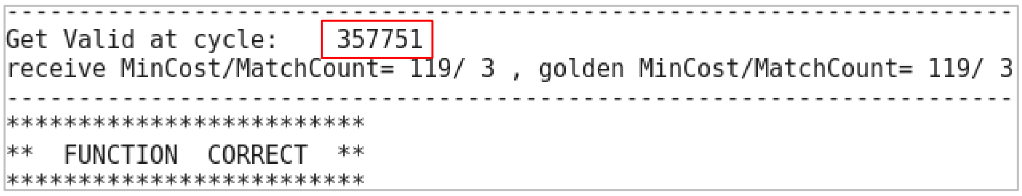
	12	22	34	54	12
	45	21	97	98	34
	54	88	21	22	34
	12	43	57	21	33
	35	98	32	1	13

3. Grading

Two design goals:

Goal 1: Total simulation cycle is smaller than 430000 cycles

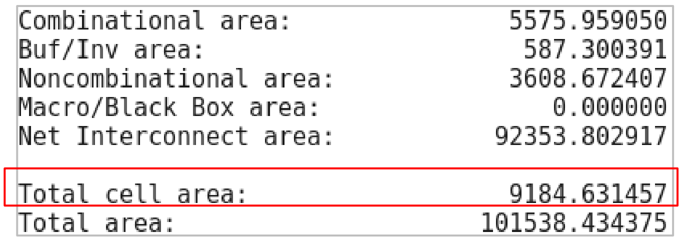
The simulation cycle is show after simulation complete, The total simulation cycle is counted by the longest cycle among the 3 patterns.



```
-------------------------------------------------------------------------
Get Valid at cycle:    357751
receive MinCost/MatchCount= 119/ 3 , golden MinCost/MatchCount= 119/ 3
-------------------------------------------------------------------------
*************************
**  FUNCTION  CORRECT  **
*************************
```

Goal 2: Area is smaller than 10000um^2 after synthesis

Design compile report area example: dc_shell>   report_area



```
Combinational area:               5575.959050
Buf/Inv area:                      587.300391
Noncombinational area:            3608.672407
Macro/Black Box area:                0.000000
Net Interconnect area:           92353.802917

Total cell area:                  9184.631457
Total area:                     101538.434375
```

The grading is classified as four levels A, B, C, D according to the completeness where A>B>C>D. The clock period is set to be 10ns and this is not adjustable by the participants.

✧ **Level A：**

a、 Gate-level and RTL simulations are completely correct given the clock period of 10ns.

b、 Goal 1 is met: total simulation cycle is smaller than 430000 cycles

c、 Goal 2 is met: Area is smaller than 10000um^2 after synthesis

Grading for level A：

According to the submission time

✧ **Level B：**

a、 Gate-level and RTL simulations are completely correct given the clock period of 10ns.

b、 Goal 1 is met: total simulation cycle is smaller than 430000 cycles

c、 Area is larger than 10000um^2 after synthesis

Grading for level B：

According to the area

✧ **Level C：**

a、 Gate-level and RTL simulations are completely correct given the clock period of 10ns.

b、 Total simulation cycle is larger than 430000 cycles

Grading for level C：

According to (synthesis area x cycle)

✧ **Level D：**

a、 RTL simulation is correct but the Gate-Level simulation is incorrect。

Grading for level C：

According to the cycle

Appendix

Appendix A、design files

1. The provided design files are listed in the following table

Table 2、design files

File	Description
JAM.v	Design files for the designers, including declarations of inputs/outputs ports.
tb.sv	Test Bench File
cost_rom	Test Pattern data
.synopsys_dc.setup synopsys_dc.setup	Initialization file for Design Compiler. Please adjust the Search Path according to the location of the Library. Noted to use the worst case library.
JAM.sdc	Design Compiler Constraint File.
report.000	Report, please see appendix C.
dc_syn.tcl	dc instructions
ncvlog.f	Ncverilog parameter files.
vcs.cmd	vcs instructions.

2. Please use JAM.v to synthesis。The Verilog module name、I/O ports are declared as follows：
The reg declaration for the output can be removed if necessary.

```
module   JAM   (
input   CLK,
input   RST,
output   reg   [2:0]   W,
output   reg   [2:0]   J,
input   [6:0]   Cost,
output   reg   [3:0]   MatchCount,
output   reg   [9:0]   MinCost,
output   reg   Valid );

endmodule
```

3. Testbench file contains define as follows

```
`define sdf_file  " ./JAM_syn.sdf "
`ifdef SDF
   initial   $sdf_annotate(`sdf_file , u_JAM) ;
`endif
```

3.1 The file name of the SDF file, please modify it according to the actual file name and path of the SDF before simulating.

3.2 The `ifdef SDF description in the testbench makes the testbench suitable for both RTL simulation and post-synthesis gate-level simulation. Competitors need to add an additional +define+SDF parameter to the simulation command when performing gate-level simulation. The example is as follows

```
ncverilog   tb.v   JAM_syn.v   -v   tsmc13_neg.v   +define+P1   +define+SDF
```

4. three sets of test samples are provided to verify the correctness of the design
please use +define+P1 、+define+P2 、+define+P3 to switch

5. Do not design for the content of these three sets of test samples, such as judging the pattern as a fixed value in the design, or judging the nth pattern to directly set the output result, etc. If found, it will not be graded.

6. The lexicographical algorithm provided is not the only method to generate full permutation. As long as the function can be completed, there is no limit to the method that must be used.

7. RTL and Gate-level simulations instructions are as follows:
RTL Simulation instructions：
RTL simulation instructions for P1 test samples

➢ ncverilog simulation instructions：
```
ncverilog   tb.v   JAM.v   +define+P1
```

➢ modelsim simulations, when compile verilog, use following instructions：
```
vlog   tb.v
```

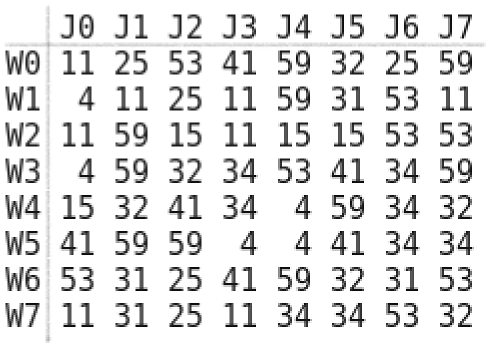
Gate-level simulation instructions:

➢ ncverilog simulation instructions：
```
ncverilog   tb.v   JAM_syn.v   -v   tsmc13_neg.v   +define+P1 +define+SDF
```
➢ For NC-Verilog, add following to output FSDB file
```
+define+FSDB   +access+r
```
modelsim users, please use built-in waveforms to debug

Appendix B、Test Pattern

Below shows the content of three test patterns

Pattern1 :



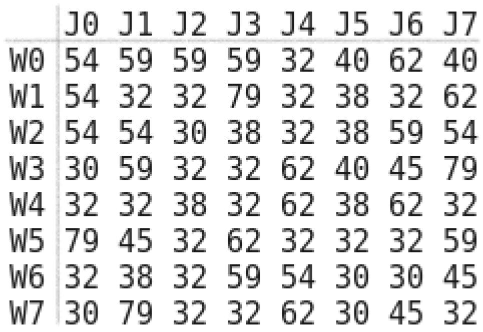
	J0	J1	J2	J3	J4	J5	J6	J7
W0	11	25	53	41	59	32	25	59
W1	4	11	25	11	59	31	53	11
W2	11	59	15	11	15	15	53	53
W3	4	59	32	34	53	41	34	59
W4	15	32	41	34	4	59	34	32
W5	41	59	59	4	4	41	34	34
W6	53	31	25	41	59	32	31	53
W7	11	31	25	11	34	34	53	32

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

Pattern2 :



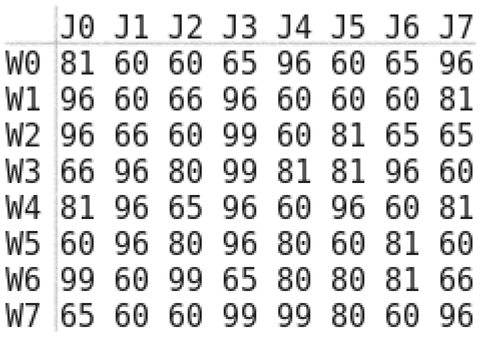
	J0	J1	J2	J3	J4	J5	J6	J7
W0	54	59	59	59	32	40	62	40
W1	54	32	32	79	32	38	32	62
W2	54	54	30	38	32	38	59	54
W3	30	59	32	32	62	40	45	79
W4	32	32	38	32	62	38	62	32
W5	79	45	32	62	32	32	32	59
W6	32	38	32	59	54	30	30	45
W7	30	79	32	32	62	30	45	32

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

Pattern3 :



	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

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

Appendix C、Grading Files

Grading files include：

(1) RTL design, RTL code for this design. Please include all the design files if they are many so that the simulation can be conducted when grading.

(2) Gate-Level design, gate-level netlist generated by the synthesizer and the corresponding SDF file；

(3) report file，each team must prepare a report.000 file according to the design. The report.000 format is shown as follows. (the extension for report file represents the version. If new version files is submitted, the new report will be report.001 and so on)

Table 3、Submitted Files

RTL category		
Design Stage	*File*	*Description*
N/A	N/A	Design Report Form
RTL Simulation	`*.v or *.vhd`	Verilog (or VHDL) synthesizable RTL code
Gate-Level category		
Design Stage	*File*	*Description*
Pre-layout Gate-level Simulation	`*_syn.v`	Verilog gate-level netlist
	`*_syn.sdf`	Pre-layout gate-level sdf

report File

FTP account:	B22xxx, FTP account
Level:	A/B/C/D design complete levels
cycle：	430000, the longest cycles among three test samples
Synthesis area：	10000，cell area
--- RTL category---	
HDL simulator :	ncverilog/vcs, HDL simulator
RTL filename :	JAM.v, RTL files
--- Pre-layout gate-level ---	
gate_level filename:	JAM_syn.v, gate-level file
gate-level sdf filename:	JAM_syn.sdf, sdf file

Appendix D、File Compression

All files listed in Table 3 need to be submitted to TSRI。 Please follow the following procedure to put all files in one folder and compress:

1. Create a result_xxx folder where "xxx" is the version。E.x. "000" is the first submission；"001"is the second submission…etc.

   ```
   > mkdir  result_000
   ```

2. Copy all files into this result_xxx folder such as:

   ```
   > cp  JAM.v       result_000
   > cp  JAM_syn.v   result_000
   > cp  report.000  result_000
     . . . . .
   ```

3. Use tar command to pack the result_xxx folder, e.g.:

   ```
   > tar  cvf  result_000.tar  result_000
   ```

   The result_000.tar is resulted

4. Use ftp to upload result_xxx.tar to ftp server of TSRI, the latest uploaded files will be graded.

   The FTP needs to be binary mode, with port:21.

   ftp account and password have been email to you。For any questions, please contact TSRI

   FTP site1：iccftp.tsri.org.tw (140.126.24.18)
   FTP site2：iccftp2.tsri.org.tw(140.110.117.9)
   EDA Cloud please see terminal message

5. If you want to upload newer version, please repeat the aforementioned step and change tar version number. You cannot change or delete or overwrite previously uploaded files.