IC Contest 2022

Table of Contents

Question:

Detail of the Question from the PDF file.

My Answer:

重點

題目要求列出所有可能情況，並找出最低成本以及最低成本的組合數量。本參考解答是利用題目上的字典排序法當作演算法作答。

首先在解題前要先了解題目意思，題目有提到:

2.3.1 JAM 電路原始成本資料的輸入

本題工作成本資料儲存在一塊 8 x 8 的同步 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，在 CLK 正緣回應 W 及 J 的內容。

JAM 電路可重覆取用 cost_rom 內資料。

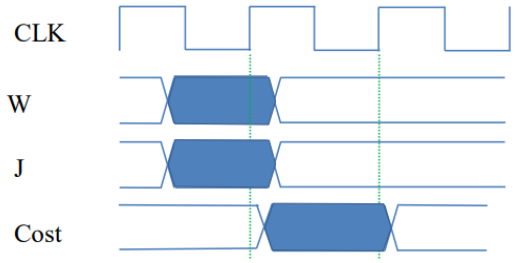


圖 3、Job Assignment Machine 之輸入訊號波形

也就是說，每一個週期只能拿到一筆資料，一次只能傳出一筆由 J,W 組合而成的數字，再收到一筆 cost 的資料。也就是說，要拿完八筆資料之後再比較資料後再判斷是否為最小成本組合。所以，每一次做上面的排序法，加上之後判斷最大最小等等運算，要在八個周期之內完成。

整體概念

整體概念是要在八個周期內每一周期輸出一個 W,J 並且累計八個周期內的 Cost 與上一周期的累計值比大小，小的留下來，計數器設為一，如果累計值一樣，計數器加一。

全部的組合情況是 8!，在最後排序到[7,6,5,4,3,2,1,0]時，下一周期要拉高 Valid 並輸出最小成本和最低成本組合，每一個組合有八個周期可以運算，所以在 RST 後最快可以完成的時間是

$$8! \times 8 = 322560$$

最快能夠在 322560 周期之後找到最小成本以及最低成本組合。

所以整個設計分為兩個區塊:

1. 設計一個電路要能夠輸出每一種序列可能，輸出最後一個序列為[7,6,5,4,3,2,1,0]時輸出一個 last 訊號，表示這是最後訊號。
2. 設計一個電路能夠累計八個周期內的 cost，並與上一個累計值比大小。

了解設計目標後，再來看如何實現細部的演算法。

字典排序法

首先，字典排序法:

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。

字典排序法一開始的組合是[0,1,2,3,4,5,6,7]，最後一組的組合是[7,6,5,4,3,2,1,0]。

第一步驟:由右邊開始，找到序列中第一組相鄰且右邊較大的位置。用七個比較器去比八個數字大小，Com[0,1], Com[1,2]…Com[6,7]，題目要求由右邊開始找比左邊大的數值，所以 Com[6,7]這一組會最優先當作替換點的比較器數值。之後再依序排下去，當發現 Com 中沒有一組為 1 時，表示序列右邊數字都比左邊數字大，就代表所有情況都排序過了。不過我是直接判斷輸出端的序列是否等於最後一組。

```
// Find switch point
genvar t;
generate
    for (t = 0; t < NUMBER_OF_WORK - 1; t = t + 1) begin : Greater_Than_Right
        assign grt[t] = list[t] < list[t+1];
    end
endgenerate
assign switch_number_index =
                        grt[6] ? 4'd7   :
                        grt[5] ? 4'd6   :
                        grt[4] ? 4'd5   :
                        grt[3] ? 4'd4   :
                        grt[2] ? 4'd3   :
                        grt[1] ? 4'd2   :
                        grt[0] ? 4'd1   :   4'd0;
```

第二步驟:找到替換點右邊數字中，比替換點大的最小數字，兩數字交換位置。這裡有兩個點要注意

1. 要是替換點右邊的數字，又同時是右邊數字中最小的數字。再比較小時，要找替換點右邊的數字，所以不是右邊的值，就把它設為最大，等等在比小時，它就不會被選到。

2. 再來是找比替換點大的最小數字，比替換點大很簡單，就直接比，但比的方式是用減法，把在右邊的值通通減去替換點的值。這樣一來比替換點的值大的數字會變成正的很小，但比替換點的值小的數字會因為 overflow 變成正的大很。

```
assign switch_val0  = (0 > (switch_number_index - 1)) ? list[0] - switch_value : 7;
assign switch_val1  = (1 > (switch_number_index - 1)) ? list[1] - switch_value : 7;
assign switch_val2  = (2 > (switch_number_index - 1)) ? list[2] - switch_value : 7;
assign switch_val3  = (3 > (switch_number_index - 1)) ? list[3] - switch_value : 7;
assign switch_val4  = (4 > (switch_number_index - 1)) ? list[4] - switch_value : 7;
assign switch_val5  = (5 > (switch_number_index - 1)) ? list[5] - switch_value : 7;
assign switch_val6  = (6 > (switch_number_index - 1)) ? list[6] - switch_value : 7;
assign switch_val7  = (7 > (switch_number_index - 1)) ? list[7] - switch_value : 7;
```

用比小的方式找到替換點右邊數字中，比替換點大的最小數字之後就把兩數交換位置。

第三步驟:把替換點後的數字順序倒過來。

設計上面的電路時，可以分很多 cycle 慢慢比較大小，不用同時在同一 cycle 做完，做計算時要注意資料之間的傳遞順序以及時序。

八個週期比較器

題目要求輸出最小成本組合數(MatchCount)，以及最小成本(MinCost)，所以需要找到最小成本的比較電路。

1. 設計一個累計八周期的累計器(Accumulator)當作目前八周期內的成本

2. 找目前為止的 MinCost，所以每一次 Accumulator 累計完就跟目前為止的 MinCost 比小:

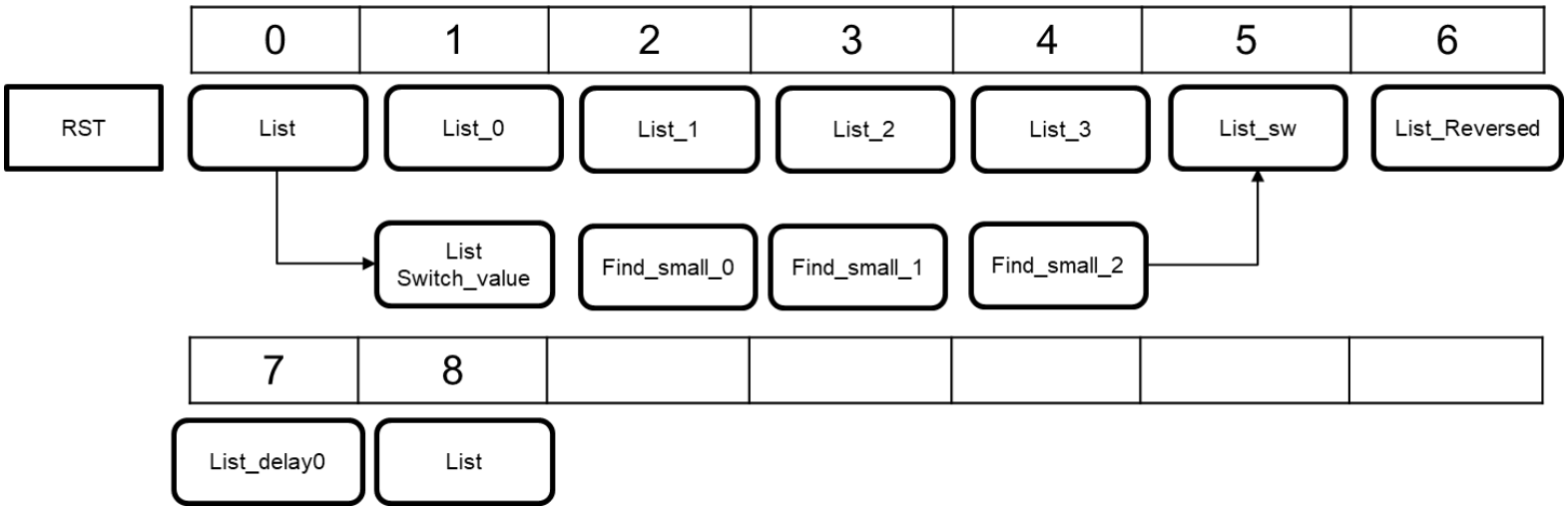
   如果 Accumulator 比較小就更新 MinCost，並將 MatchCount 設為 1

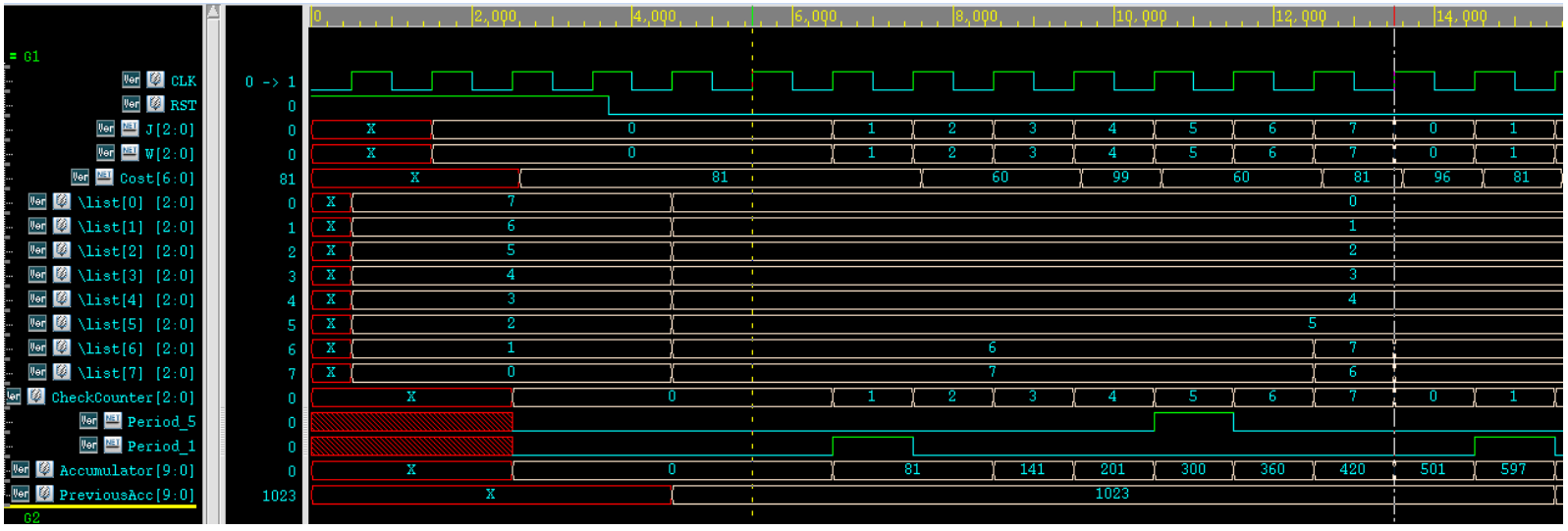
   如果 Accumulator 一樣，MatchCount+1

設計時注意，累計八個週期的 Accumulator，再要被更新值之前，要將 Accumulator 存到另一個暫存器 PreviousAcc，因為如果在拿到八個周期內的累計值時比大小，會因為下一筆 cost 又進來，所以 Accumulator 被更新為新一筆 Cost 而非上八周期內的累計值。

Data Flow Graph

1. List 為 input 也是 output，在傳輸到第八 cycle 時要回傳數值，當作下一次開始的序列。
2. 我是分成三個 cycle 來找最小值，反正 cycle 數很多，慢慢比。
3. 由於分三個 cycle 在比大小，原本的數值也要儲存，這樣在字典排序法第二步驟，要交換數字時才能知道要跟誰換。
4. 換完下一 cycle 把替換點後的數字反過來。就是下一序列的開始。
5. 由於要八個 cycle 當作一周期，算完沒事還要延遲一周期捕到八周期。





RTL sim Result:

```
*******************************
** Simulation Start          **
*******************************
PATTERN:   1
-------------- Cost Table --------------
Jobs       0   1   2   3   4   5   6   7
worker0:  11  25  53  41  59  32  25  59
worker1:   4  11  25  11  59  31  53  11
worker2:  11  59  15  11  15  15  53  53
worker3:   4  59  32  34  53  41  34  59
worker4:  15  32  41  34   4  59  34  32
worker5:  41  59  59   4   4  41  34  34
worker6:  53  31  25  41  59  32  31  53
worker7:  11  31  25  11  34  34  53  32
------------------------------------------------------------------------
Get Valid at cycle:    322576
receive MinCost/MatchCount=  119/ 3 , golden MinCost/MatchCount= 119/ 3
------------------------------------------------------------------------
************************
**  FUNCTION  CORRECT  **
************************


*******************************
** Simulation Start          **
*******************************
PATTERN:   2
-------------- Cost Table --------------
Jobs       0   1   2   3   4   5   6   7
worker0:  54  59  59  59  32  40  62  40
worker1:  54  32  32  79  32  38  32  62
worker2:  54  54  30  38  32  38  59  54
worker3:  30  59  32  32  62  40  45  79
worker4:  32  32  38  32  62  38  62  32
worker5:  79  45  32  62  32  32  32  59
worker6:  32  38  32  59  54  30  30  45
worker7:  30  79  32  32  62  30  45  32
------------------------------------------------------------------------
Get Valid at cycle:    322576
receive MinCost/MatchCount=  250/ 6 , golden MinCost/MatchCount= 250/ 6
------------------------------------------------------------------------
************************
**  FUNCTION  CORRECT  **
************************

Simulation complete via $finish(1) at time 3225755 NS + 0
../tb.sv:194                    $finish;
```

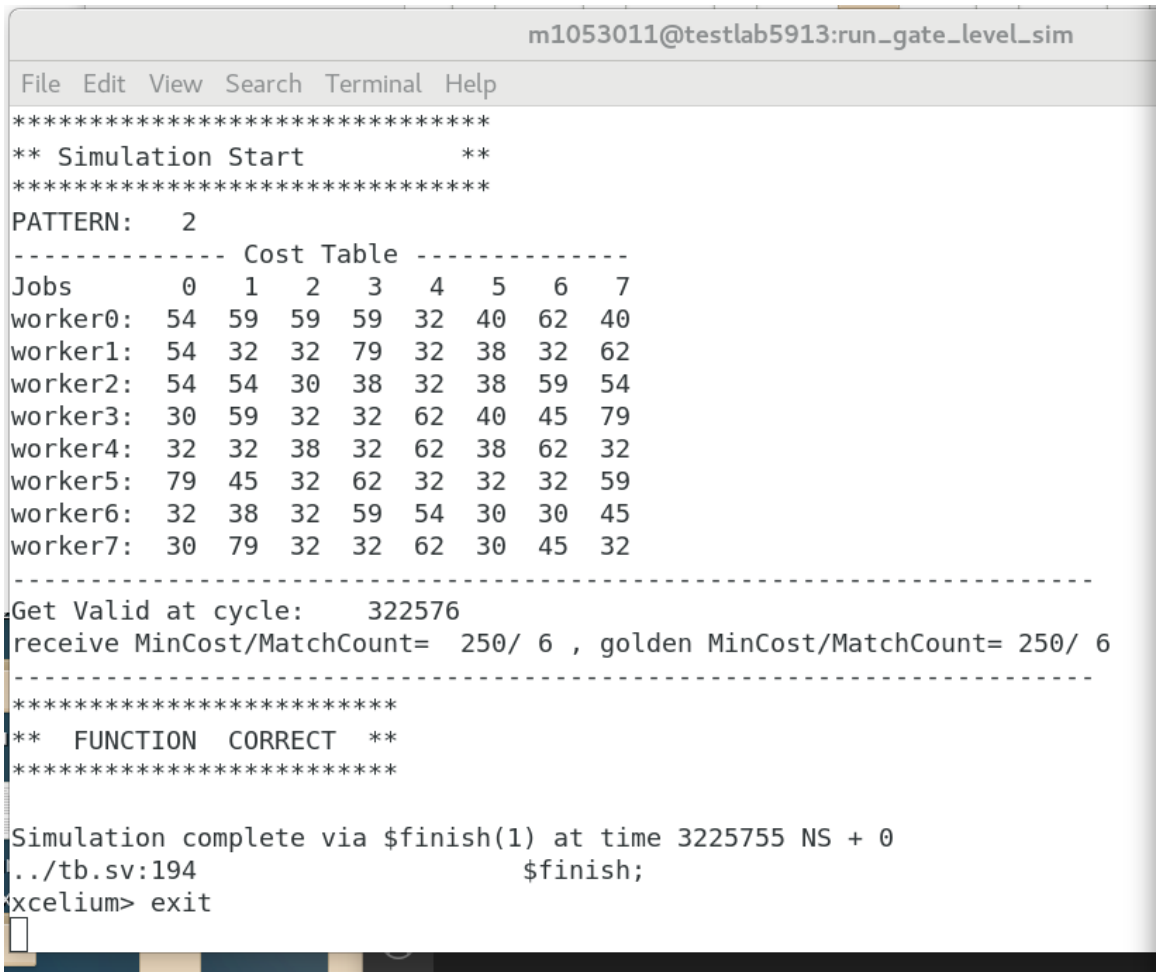
```
*******************************
** Simulation Start          **
*******************************
PATTERN:   3
-------------- Cost Table --------------
Jobs       0   1   2   3   4   5   6   7
worker0:  81  60  60  65  96  60  65  96
worker1:  96  60  66  96  60  60  60  81
worker2:  96  66  60  99  60  81  65  65
worker3:  66  96  80  99  81  81  96  60
worker4:  81  96  65  96  60  96  60  81
worker5:  60  96  80  96  80  60  81  60
worker6:  99  60  99  65  80  80  81  66
worker7:  65  60  60  99  99  80  60  96
--------------------------------------------------------------------------
Get Valid at cycle:    322576
receive MinCost/MatchCount=  485/ 9 , golden MinCost/MatchCount= 485/ 9
--------------------------------------------------------------------------
************************
**  FUNCTION  CORRECT  **
************************

Simulation complete via $finish(1) at time 3225755 NS + 0
../tb.sv:194                    $finish;
xcelium> exit
```

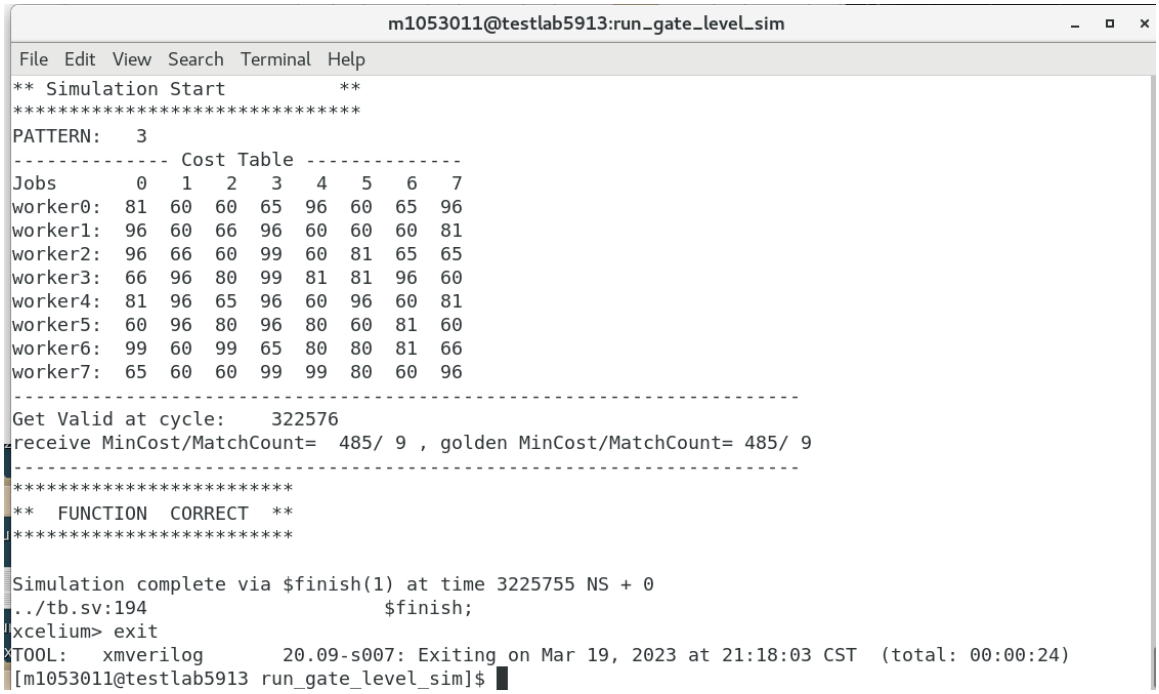
Design Compiler Area

```
Library(s) Used:

    slow (File: /users/m1053011/Desktop/verilog_sim/

Number of ports:                          130
Number of nets:                          1538
Number of cells:                         1402
Number of combinational cells:            981
Number of sequential cells:               407
Number of macros/black boxes:               0
Number of buf/inv:                        226
Number of references:                       2

Combinational area:               7838.593151
Buf/Inv area:                     1137.257986
Noncombinational area:           10820.925177
Macro/Black Box area:                0.000000
Net Interconnect area:          166367.529205

Total cell area:                 18659.518327
Total area:                     185027.047533
1
```

Gate-Level Sim Results

```
m1053011@testlab5913:run_gate_level_sim
File  Edit  View  Search  Terminal  Help
*******************************
** Simulation Start          **
*******************************
PATTERN:   1
-------------- Cost Table --------------
Jobs       0   1   2   3   4   5   6   7
worker0:  11  25  53  41  59  32  25  59
worker1:   4  11  25  11  59  31  53  11
worker2:  11  59  15  11  15  15  53  53
worker3:   4  59  32  34  53  41  34  59
worker4:  15  32  41  34   4  59  34  32
worker5:  41  59  59   4   4  41  34  34
worker6:  53  31  25  41  59  32  31  53
worker7:  11  31  25  11  34  34  53  32
---------------------------------------------------------------------
Get Valid at cycle:    322576
receive MinCost/MatchCount=  119/ 3 , golden MinCost/MatchCount= 119/ 3
---------------------------------------------------------------------
************************
**  FUNCTION  CORRECT  **
************************

Simulation complete via $finish(1) at time 3225755 NS + 0
../tb.sv:194                    $finish;
xcelium> exit
```


m1053011@testlab5913:run_gate_level_sim
File Edit View Search Terminal Help
****
** Simulation Start **
****
PATTERN: 2
---------- Cost Table ----------
Jobs 0 1 2 3 4 5 6 7
worker0: 54 59 59 59 32 40 62 40
worker1: 54 32 32 79 32 38 32 62
worker2: 54 54 30 38 32 38 59 54
worker3: 30 59 32 32 62 40 45 79
worker4: 32 32 38 32 62 38 62 32
worker5: 79 45 32 62 32 32 32 59
worker6: 32 38 32 59 54 30 30 45
worker7: 30 79 32 32 62 30 45 32
Get Valid at cycle: 322576
receive MinCost/MatchCount= 250/ 6 , golden MinCost/MatchCount= 250/ 6
****
** FUNCTION CORRECT **
****
Simulation complete via $finish(1) at time 3225755 NS + 0
../tb.sv:194 $finish;
xcelium> exit


m1053011@testlab5913:run_gate_level_sim
File Edit View Search Terminal Help
** Simulation Start **
****
PATTERN: 3
---------- Cost Table ----------
Jobs 0 1 2 3 4 5 6 7
worker0: 81 60 60 65 96 60 65 96
worker1: 96 60 66 96 60 60 60 81
worker2: 96 66 60 99 60 81 65 65
worker3: 66 96 80 99 81 81 96 60
worker4: 81 96 65 96 60 96 60 81
worker5: 60 96 80 96 80 60 81 60
worker6: 99 60 99 65 80 80 81 66
worker7: 65 60 60 99 99 80 60 96
Get Valid at cycle: 322576
receive MinCost/MatchCount= 485/ 9 , golden MinCost/MatchCount= 485/ 9
****
** FUNCTION CORRECT **
****
Simulation complete via $finish(1) at time 3225755 NS + 0
../tb.sv:194 $finish;
xcelium> exit
TOOL: xmverilog 20.09-s007: Exiting on Mar 19, 2023 at 21:18:03 CST (total: 00:00:24)
[m1053011@testlab5913 run_gate_level_sim]$