Design: Three-Inning Baseball Game

Data Preparation

1. Extract test data from TA's directory:
   % tar -xvf ~iclabTA01/Lab02.tar
2. The extracted Lab02/ directory contains:
   a. **Exercise** (You'll need to work on it.)
   b. Practice

Design Description

In this Lab, students will design a simplified baseball game system. The game consists of three innings, simulating common batter actions such as hits, home runs, and walks. Based on the batter's actions, the base runners' status will change, which in turn affects the score. Students need to implement a state machine to track the progress of each inning, including the top and bottom halves, and record the scores for each team. At the end of the game, the system will output the final results, determining which team wins or if the game is tied.

Game Rules

On a baseball field, there are four bases: home plate, first base, second base, and third base. After the batter hits the ball, depending on the result of the hit, the batter will run towards first base, and based on the running status and the outcome of the next hit, may advance to second base, third base, and finally return to home plate to score a run.

- **Scoring**: A run is scored when the batter or runner returns to home plate.
- **Outs**: A batter or runner is declared out when they fail to reach the base safely. Each team has three outs per half-inning. After three outs, the offense ends, and the teams switch sides.
- **Top and Bottom of the Inning**: The guest team (Team A) bats in the top half of the inning, while the home team (Team B) bats in the bottom half.
- **Three-Inning Game**: After three innings, the scores of both teams will be compared to determine the winner.
- If Team A is still trailing after the top of the 3rd inning, the game continues into the bottom of the 3rd inning, but **Team B's score in the bottom half will not count**, as they have already secured victory.
- If there is no decisive result after the top of the 3rd inning (i.e., a tie or Team A leading), the bottom of the 3rd inning will still be played until three outs, even if Team B secures a comeback victory.

The action signal in the input represents the batter's behavior, with eight possible actions:

- Walk (3'd0)
- Single (3'd1)
- Double (3'd2)
- Triple (3'd3)
- Home Run (3'd4)
- Bunt (3'd5)
- Ground Ball (3'd6)
- Fly Ball (3'd7)

Common Keywords:

- **Advance**: Moving from 1B to 2B, 2B to 3B, or 3B to home plate.
- Hit and Run (**Run Early**): With 2 outs, runners will attempt to score regardless of the outcome of the hit, ignoring the outs.
- **Change of Sides**: After three outs, the half-inning ends, and the new half-inning starts with zero outs, with the teams switching sides (offense/defense).

Effects of Each Action:

- **Walk (action = 3'd0):**

  Batter: advances to 1B.

  Runners: Advance in order (if there is no runner ahead of them).
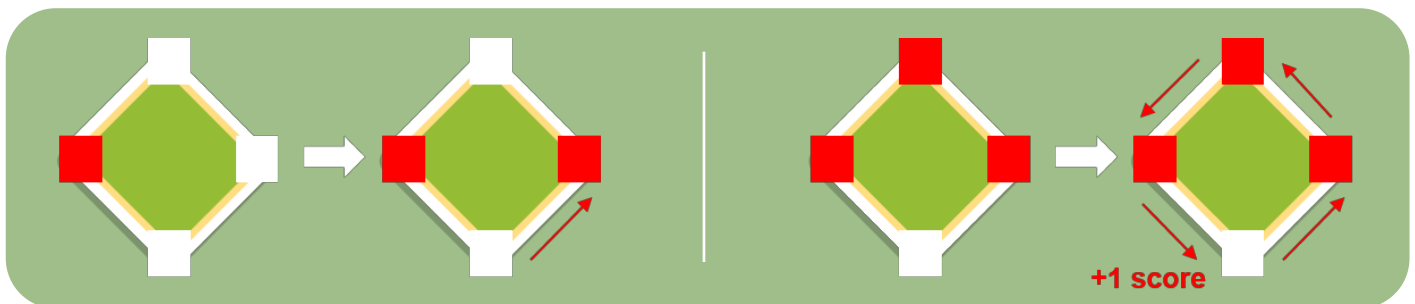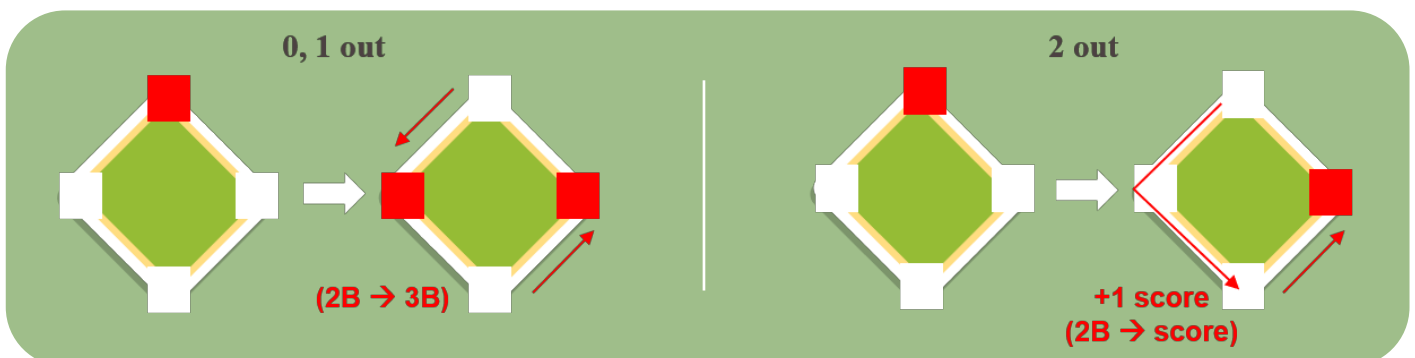

- **Single (action = 3'd1):**

  Batter: advances to 1B.

  Runners:

  - 0 or 1 out: 1B → 2B, 2B → 3B, 3B → scores.
  - 2 outs: "Runners" run early, and advance an extra base. 1B → 3B, 2B → scores, 3B → scores.
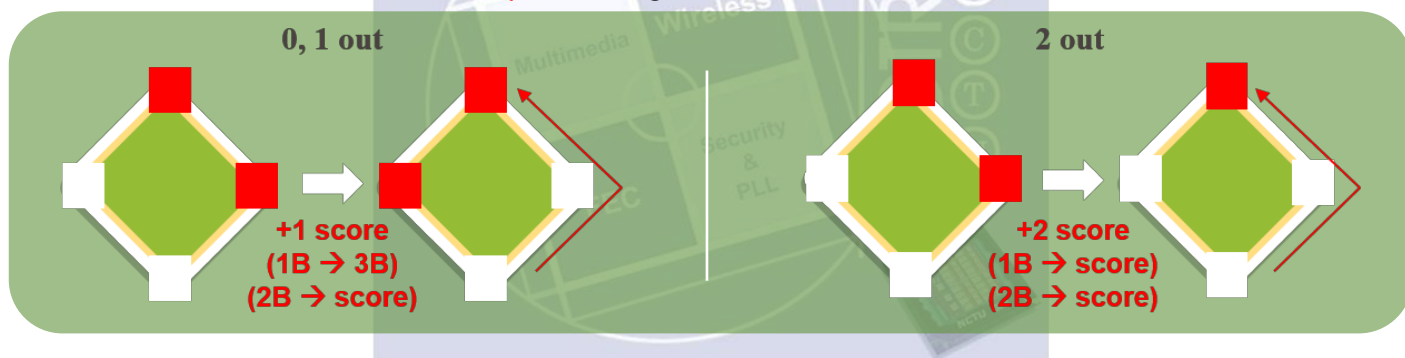
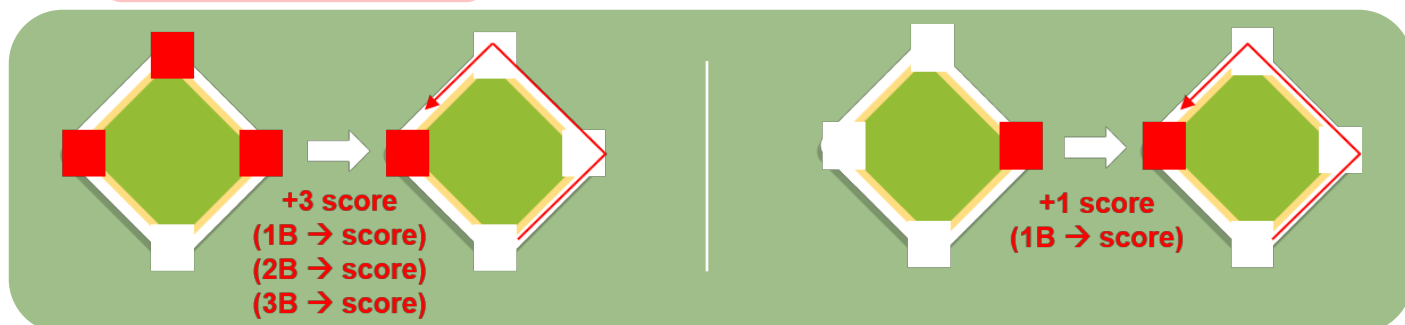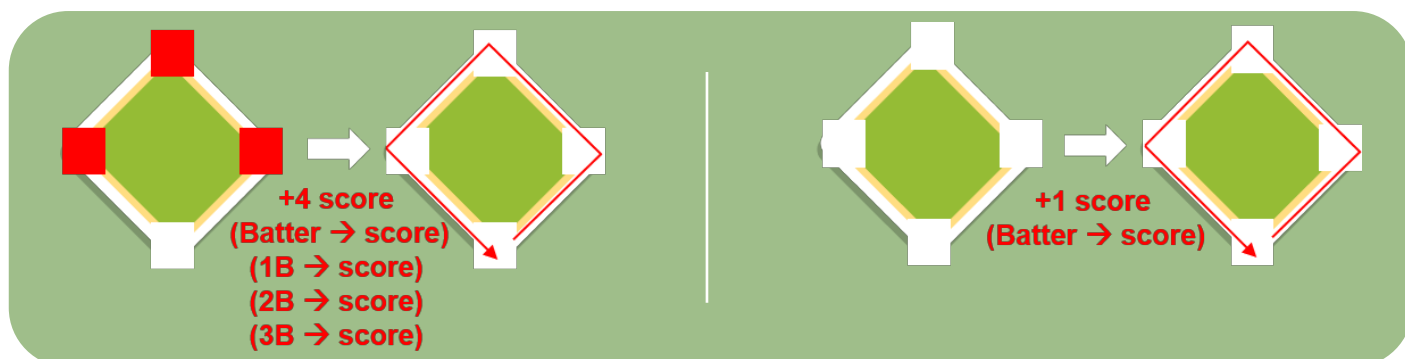- **Double (action = 3'd2):**

  Batter: advances to 2B.

  Runners:

  - 0 or 1 out: 1B → 3B, 2B → scores, 3B → scores.

  - 2 outs: "Runners" run early, advancing an extra base. 1B → scores, 2B → scores, 3B → scores.

0, 1 out

+1 score
(1B → 3B)
(2B → score)

2 out

+2 score
(1B → score)
(2B → score)

- **Triple (action = 3'd3):**

  Batter: advances to 3B.

  Runners: All runners score.

+3 score
(1B → score)
(2B → score)
(3B → score)

+1 score
(1B → score)

- **Home Run (action = 3'd4):**

  Batter: scores.

  Runners: All runners score.

+4 score
(Batter → score)
(1B → score)
(2B → score)
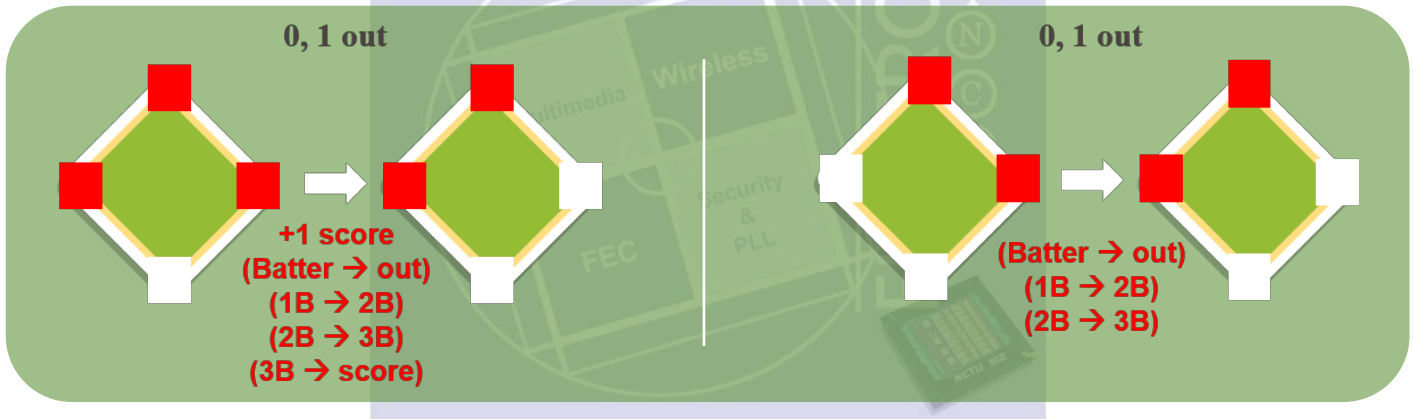(3B → score)

+1 score
(Batter → score)

- **Bunt (action = 3'd5):**

 **This action will only occur in patterns with 0 or 1 out and runners on base.**

  Batter: out.

  Runners: Advance to the next base at the same time.

- **Ground Ball (action = 3'd6):**

  - 0 outs: If there is a runner on 1B, it results in a Double Play (DP, +2outs), with the batter and the runner on 1B out, and other runners advance. If 1B is empty, only the batter is out, and the other runners advance.

  - 1 out: If there is a runner on 1B, it results in a Double Play (DP, +2outs), leading to 3 outs and the end of the half-inning. If 1B is empty, only the batter is out, and other runners advance.

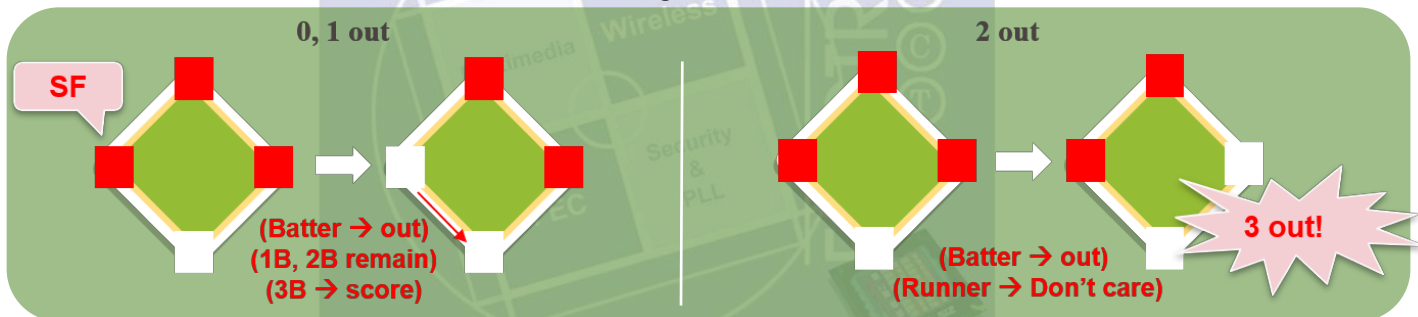 - 2 outs: The batter is out, and the half-inning ends.
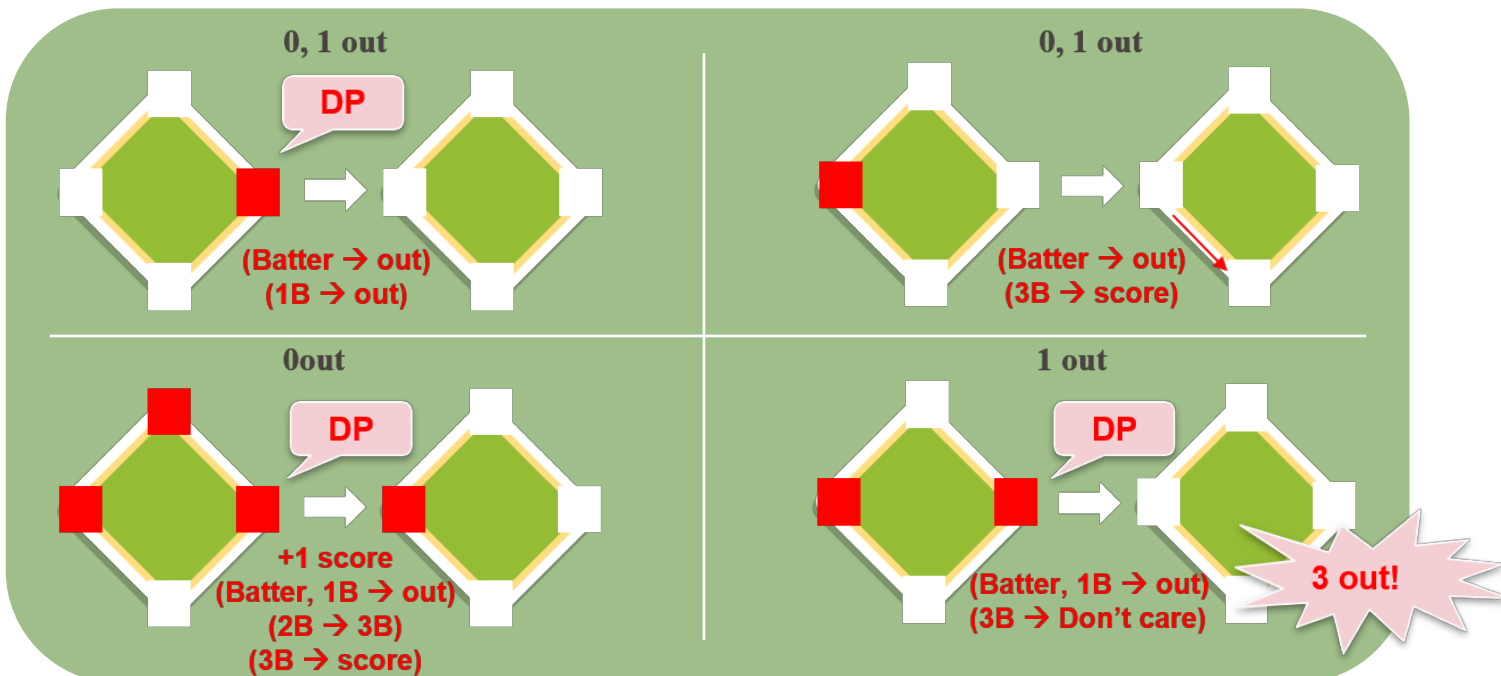
- **Fly Ball (action = 3'd7):**
  - ◆ 0 or 1 out: If there is a runner on 3B, it triggers a sacrifice fly (SF), the batter is out, and the runner on 3B scores. Other runners remain on their bases. If there is no runner on 3B, only the batter is out, and the other runners remain in place. （只有 3 壘能跑）
  - ◆ 2 outs: The batter is out, and the half-inning ends.

Important Notes after Reading the Input:

- 3 outs take precedence: Even if a ground ball advances the runner on 3B or a fly ball in a 2-out situation with a runner on 3B, the 3-out rule has higher priority over scoring.
- Bunt, ground ball, and fly ball will always result in an out.
- Patterns will not consider the status of the bases, but all bases must be cleared when switching sides.

I/O specification

Signals name	Port	Bit Width	Definition
clk	input	1	Clock signal
rst_n	input	1	Asynchronous active-low reset
in_valid	input	1	High when input signals are valid
inning	input	2	Current inning number (1 for 1st inning, 2 for 2nd inning, etc.).
half	input	1	Indicates whether it's the top (0) or bottom (1) of the inning.
action	input	3	Action code representing the batter's action (e.g., walk, single, home run).
out_valid	output	1	High when output is valid.
score_A	output	8	The score of Team A (guest team).
score_B	output	8	The score of Team B (home team).
result	output	2	Team A wins: 0 Team B wins: 1 Tie: 2

Specifications

1. Top module name: BB (Filename: BB.v)
2. It is an asynchronous reset and active-low architecture. If you use synchronous reset (reset after clock starting) in your design, you may fail to reset signals.
3. Verilog commands *//synopsys dc_script_begin, //synopsys dc_script_end, //synopsys translate_off, //synopsys translate_on*, are only allowed during the usage of including and setting designware IPs, other design compiler optimizations are forbidden.

4. The clock period of the design is fixed to 10ns.
5. The next group of inputs will come in 10 cycles after your out_valid pull down.
6. The synthesis result of data type cannot include any LATCH.
7. After synthesis, you can check BB.area and BB.timing in the folder "Report".
8. The slack in the timing report should be non-negative and the result should be MET.
9. The gate level simulation cannot include any timing violation.
10. The latency of your design in each pattern should not be larger than 100 cycles. The latency is the clock cycles between the falling edge of the **in_valid** and the rising edge of the **out_valid**.
11. Don't use any wire/reg/submodule/parameter name called error, congratulation, latch or fail otherwise you will fail the lab.
   Note: means any char in front of or behind the word. e.g: error_note is forbidden.
12. The out_valid cannot overlap in_valid.
13. Don't write chinese comments or other language comments in the file you turned in.
14. Any form of display or printing information in verilog design is forbidden. You may use this methodology during debugging, but the file you turn in should not contain any coding that is not synthesizable.

Block Diagram
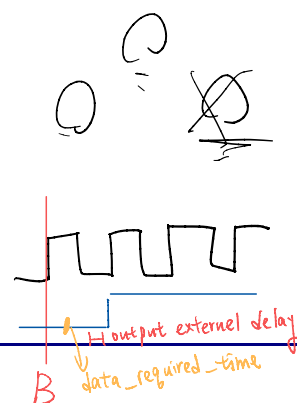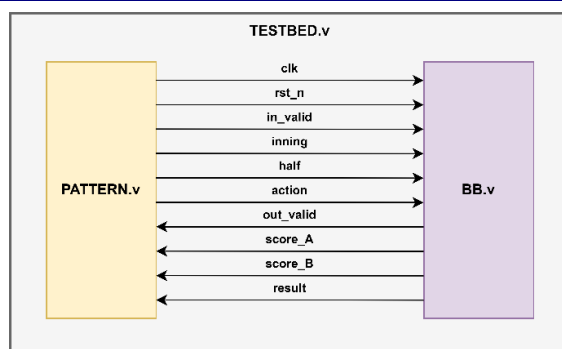
Note

1. **Grading policy:**
   RTL and gate-level simulation correctness: 70%
   Performance (Area * Execution Cycle): 30%
2. **Please submit your design through Lab02/Exercise/09_SUBMIT**
   - 1st_demo   deadline: 2024/09/23(Mon.) 12:00:00
   - 2nd_demo   deadline: 2024/09/25(Wed.) 12:00:00
   - If uploaded files violate the naming rule, you will get 5 deduct points.
3. **Template folders and reference commands:**
   |-- **01_RTL**
   | |-- **./01_run_vcs_rtl**   ➔ RTL simulation
   |-- **02_SYN**
   | |-- **./01_run_dc_shell** ➔ Synthesis
   | |-- **./08_check**   ➔ Check design's timing and if there's latch or not in *syn.log*)
   |-- **03_GATE**
   | |-- **./01_run_vcs_gate** ➔ Gate-level simulation
   `-- **09_SUBMIT**
      |-- **./00_tar**   ➔ Tar all your design
      |-- **./01_submit**   ➔ Submit files
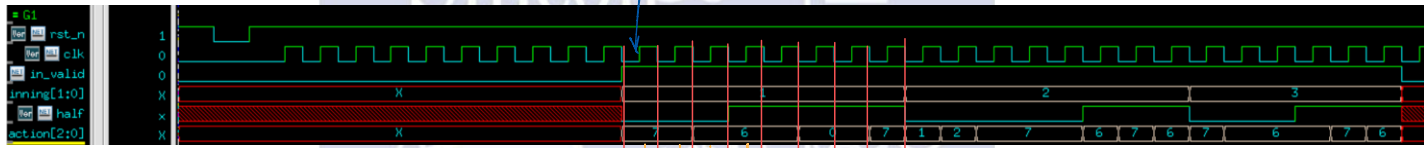 |-- **./02_check**   ➔ Check files
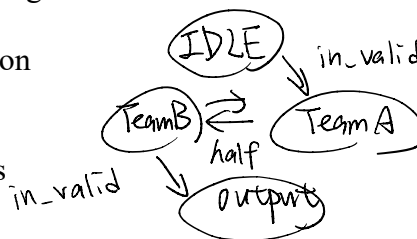
1. how to store status of base? 3 bit   4. half != half_reg

2. how to store number of outs? 2 bit   5. inning == 3, half == out ==

3. just if-else   ⇒ out_valid

Sample Waveform

- **Input signal**

@posedge(clk)

first store to get full cycles

fly fly Gnd Gnd Gnd walk walk fly

- **Output signal**
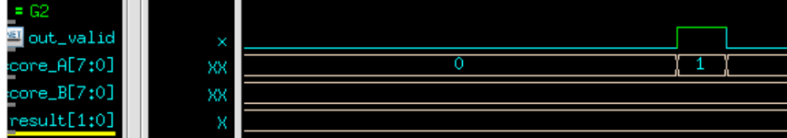
Sample File Content

- **input.txt**

```
1   #1
2   -1
3   7
4   7
5   6
6   +1
7   6
8   6
9   0
10  0
11  7
12  -2
13  1
14  2
15  7
16  7
17  7
18  +2
19  6
20  7
21  6
22  -3
23  7
24  6
25  6
26  +3
27  6
28  7
29  6
30  ===
```

- **input_cn.txt**

```
1   #1（第01場比賽）
2   -1（第1局上半）
3   7（行動：高飛球，壘包狀態：1B=無，2B=無，3B=無，出局數：1，進攻隊伍本局得分：0）
4   7（行動：高飛球，壘包狀態：1B=無，2B=無，3B=無，出局數：2，進攻隊伍本局得分：0）
5   6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：3，進攻隊伍本局得分：0）
6   分數：0：0
7   +1（第1局下半）
8   6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：1，進攻隊伍本局得分：0）
9   6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：2，進攻隊伍本局得分：0）
10  0（行動：四死球，壘包狀態：1B=有，2B=無，3B=無，出局數：2，進攻隊伍本局得分：0）
11  0（行動：四死球，壘包狀態：1B=有，2B=有，3B=無，出局數：2，進攻隊伍本局得分：0）
12  7（行動：高飛球，壘包狀態：1B=有，2B=有，3B=無，出局數：3，進攻隊伍本局得分：0）
13  分數：0：0
14  -2（第2局上半）
15  1（行動：1H，壘包狀態：1B=有，2B=無，3B=無，出局數：0，進攻隊伍本局得分：0）
16  2（行動：2H，壘包狀態：1B=無，2B=有，3B=有，出局數：0，進攻隊伍本局得分：0）
17  7（行動：高飛球，壘包狀態：1B=無，2B=有，3B=無，出局數：1，進攻隊伍本局得分：1）
18  7（行動：高飛球，壘包狀態：1B=無，2B=有，3B=無，出局數：2，進攻隊伍本局得分：1）
19  7（行動：高飛球，壘包狀態：1B=無，2B=有，3B=無，出局數：3，進攻隊伍本局得分：1）
20  分數：1：0
21  +2（第2局下半）
22  6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：1，進攻隊伍本局得分：0）
23  7（行動：高飛球，壘包狀態：1B=無，2B=無，3B=無，出局數：2，進攻隊伍本局得分：0）
24  6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：3，進攻隊伍本局得分：0）
25  分數：1：0
26  -3（第3局上半）
27  7（行動：高飛球，壘包狀態：1B=無，2B=無，3B=無，出局數：1，進攻隊伍本局得分：0）
28  6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：2，進攻隊伍本局得分：0）
29  6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：3，進攻隊伍本局得分：0）
30  分數：1：0
31  +3（第3局下半）
32  6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：1，進攻隊伍本局得分：0）
33  7（行動：高飛球，壘包狀態：1B=無，2B=無，3B=無，出局數：2，進攻隊伍本局得分：0）
34  6（行動：滾地球，壘包狀態：1B=無，2B=無，3B=無，出局數：3，進攻隊伍本局得分：0）
35  分數：1：0
36  分數：1：0，A隊贏球！
37  ===
```

- **output.txt**

```
1   #1
2   1
3   0
4   0
```