NYCU EE IC LAB - SPRING 2022

Lab02 Exercise

Design: Digit matching machine

Data Preparation

1. Extract test data from TA's directory:

   % tar xvf ~iclabta01/Lab02.tar

2. The extracted LAB directory contains:

   a. Practice/

   b. Exercise/

Design Description

■ ***Bulls and Cows***

Bulls and Cows (also known as "Cows and Bulls" or "Pigs and Bulls") is a classical code-breaking mind or paper and pencil game for two or more players, which is similar to the web-based word game "*Wordle*" released in 2021.

The numerical version of the game is usually played with 4 digits, but can played with any number of digits. The player is given a 4-digit secret number, the digits must be all different, then try to guess each digit. If the matching digits are in their right positions, they are "bulls" or "A", if in different positions, they are "cows" or "B". For example:

- Secret number: 4271
- Player's try: 1234
- Result: 1 bull and 2 cows, or 1A2B. (The bull is "2", the cows are "4" and "1".)

Based on the result, the player continues the next try and finally reveals the secret number.

■ *Brief Introduction*

According to the above game, in this Exercise, you need to design a **digit matching machine**. In each round, you are given four things, an **8-digit keyboard**, a **5-digit secret answer**, a **set of weights** and a **match target**.

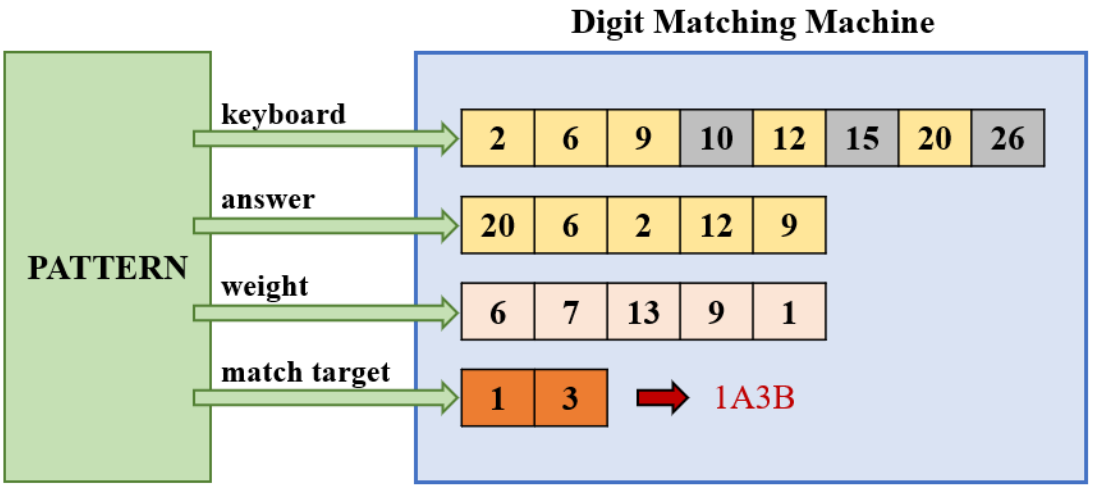
The summary of these four input information are as following:

Information	Digit number	Description
keyboard	8	All 8 digits are nonzero positive number in the range of 1 to 31, and one digit only appear one time.
secret answer	5	All 5 digits of the secret answer are chosen from the keyboard, and one digit is only chosen one time.
weight	5	All 5 digits are nonzero positive number in the range of 1 to 15, and one digit only appear one time.
match target	2	The match target is the requirement such as 1A2B, 3A0B, etc.

After acquiring the information, your design should reorder the 8 digits from the keyboard to generate **5-digit combinations**, then finally output the one that **meets the match target by comparing to the secret answer** and **has the maximum weighted score simultaneously**.

■ *Example*

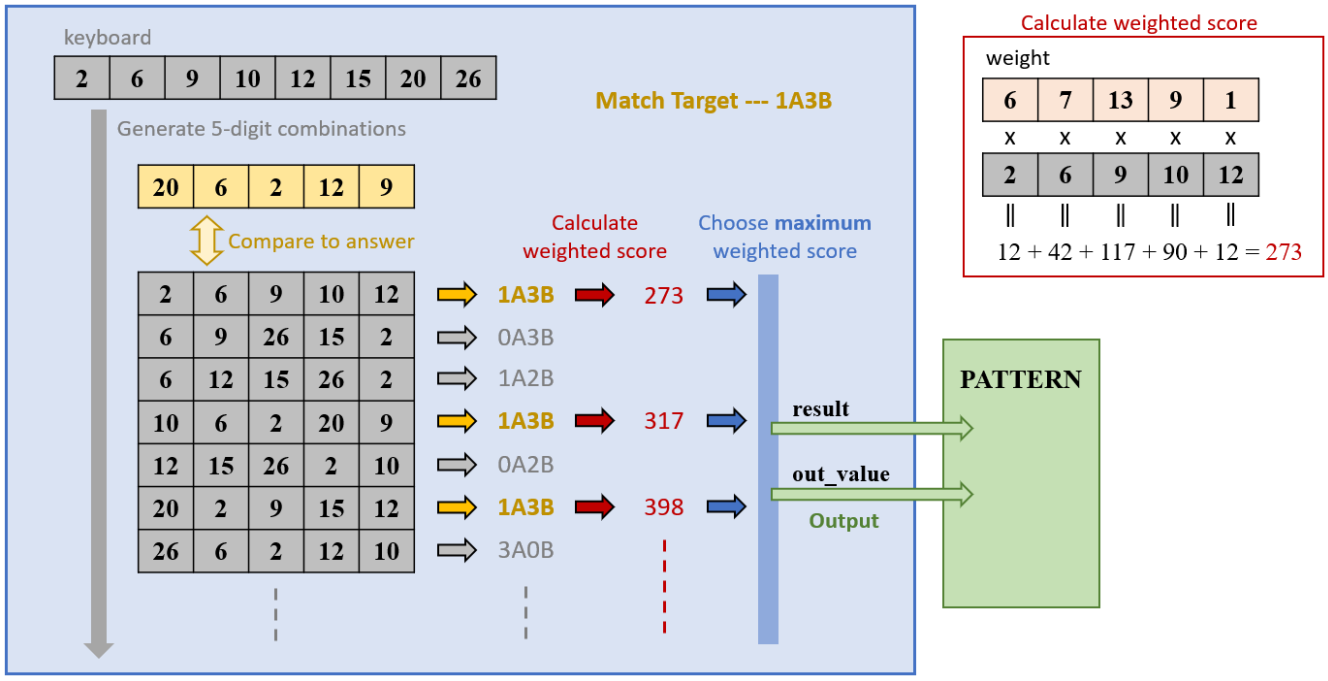
- Step-1: Input keyboard, secret answer, weight set and the match target.



  - The 8 digits of keyboard are sorted from smallest to largest.
  - Both 5 digits of answer and weight are arranged randomly without sorting.

- Step-2: Reorder the 8 digits from the keyboard to generate 5-digit combinations, then find which can meet the match target and calculate the corresponding weighted score.

- Step-3: Usually, there are multiple combinations that can all meet the match target, output the one has the maximum weighted score.

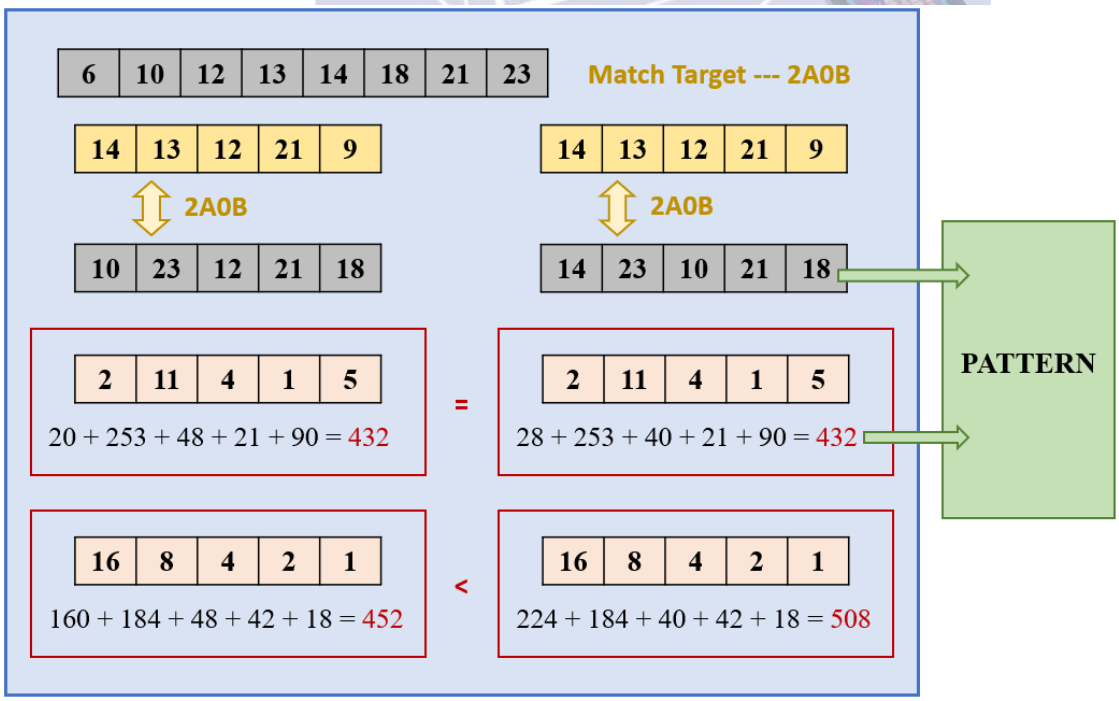


■ ***Supplement***

- Corner case

If there are more than one **matching combinations** that have **same maximum weighted score**, output the one which has the higher weighted score with the weight set below.

16	8	4	2	1



Inputs

■ The following are the definitions of input signals

Input Signals	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when input signals are valid.
keyboard	5	The digit basis.
answer	5	The correct answer.
weight	4	The weight set for each digit position.
match_target	3	The matching requirement.

1. All input signals are synchronized at **negative edge** of the clock.
2. When **in_valid** is low, input is tied to unknown state.

Outputs

■ The following are the definitions of output signals

Output Signals	Bit Width	Definition
out_valid	1	High when output is valid.
result	5	Output the result 5-digit combination.
out_value	11	Output the corresponding maximum weighted score.

1. All outputs should be low after **rst_n** assert.
2. Output signals should be synchronized at clock positive edge.
3. TA's pattern will capture your output for checking at **negative** clock edge.
4. All operations are unsigned.

Specifications

Top module

1. Top module name: WD (File name: WD.v)

Reset

2. It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) in your design, you may fail to reset signals.
3. The reset signal(**rst_n)** would be given only once at the beginning of simulation. All output signals should be reset after the reset signal is asserted.

Design Constraints

4. The clock period is set to 7ns, and you cannot change it.
5. The execution latency is limited in 20000 cycles. The latency is the clock cycles between the falling edge of the last cycle of **in_valid** and the rising edge of the **out_valid**.

out valid

6. The **result** and **out_value** should be reset after your **out_valid** is pulled down.
7. The **out_valid** is limited to be high for only 5 cycles when the output value is valid.
8. The next input pattern will be triggered **2~4 negative edge of clock** after your **out_valid** falls.
9. The **out_valid** should not be high when **in_valid is** high.
10. The **result** and **out_value** should be correct when **out_valid** is high.

Synthesis

11. The input delay is set to 0.5*(clock period).
12. The output delay is set to 0.5*(clock period), and the output loading is set to 0.05.
13. The synthesis result of data type cannot include any LATCH.
14. After synthesis, the area report is valid only when the slack in the end of timing report should be non-negative and the result should be MET.

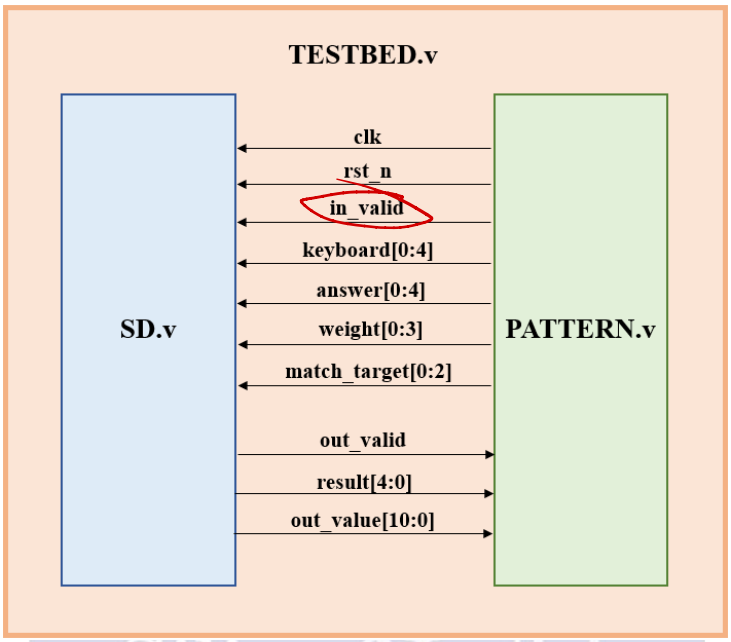
Gate level simulation

15. The gate level simulation cannot include any timing violations without the *notimingcheck* command.

Supplement

16. 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.
17. Don't write chinese comments or other language comments in the file you turned in.
18. 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.
19. Using the above commands are allowed, however any error messages during synthesize and simulation, regardless of the result will lead to failure in this lab.
20. 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:**
   - Function Validity: 70% (RTL and Gate-level simulation correctness)
   - Performance: 30% (Area * Total latency time)
   - The performance is determined by area and latency of your design. The less cost your design has, the higher grade you get.
   - The grade of 2nd demo would be 30% off.
2. **Please upload the following file on new e3 platform before 23:59 p.m. on Mar. 13:**
   - WD_iclabxxx.v (xxx is your account number, i.e. WD_iclab999.v)
   - If the uploaded file violating the naming rule, you will get 5 deduct points.
3. **Template folders and reference commands:**

   01_RTL/ (RTL simulation) **./01_run**

   02_SYN/ (Synthesis) **./01_run_dc**

   (Check **latch** by searching the keyword "**Latch**" in **syn.log**)

   (Check the design's timing in /Report/**WD.timing**)

   (Check the design's area in /Report/**WD.area**)

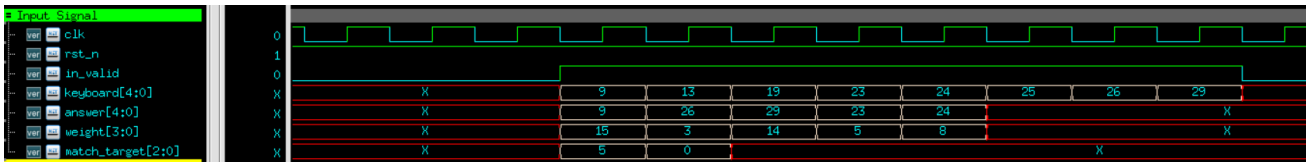
   03_GATE / (Gate-level simulation) **./01_run**

- You can key in **./09_clean_up** to clear all log files and dump files in each folder.

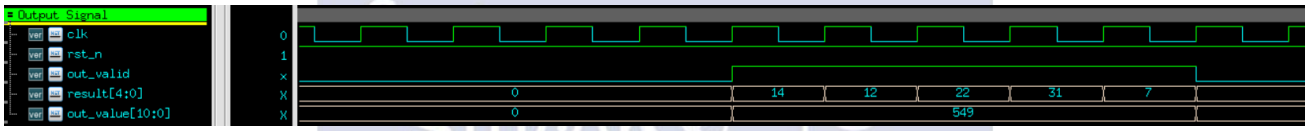
Sample Waveform

1. Asynchronous reset and active-low and reset all output

2. 8 cycles for input the information of each round(pattern)



3. 5 cycles for output the result of each round(pattern)



- Note that **out_value** will be checked at the fifth cycle after your **out_valid** is pulled up.