NCTU-EE IC LAB – Fall2021

Lab02 Exercise

Design: Knight's Tour

Data Preparation

1. Extract test data from TA's directory:

   % tar -xvf ~iclabta01/Lab02.tar

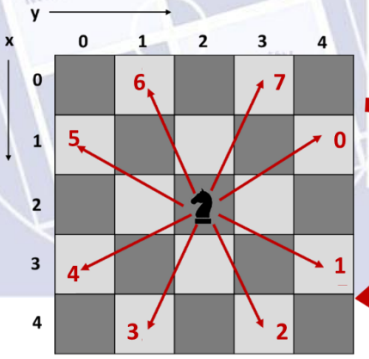
2. The extracted LAB director contains:

   a. Practice

   b. Exercise

Design Description

The Knight's Tour is a problem of moving a knight on a chessboard, and the knight visits every square **exactly once**. In every move, the knight has 8 moving directions. In this exercise, you will get a halfway toured chessboard with the number of knight's moves. In the mean time you will get the priority of knight's moving strategy. For example, if the priority is 0 that means you should choose the direction 0 first for each step, in case the direction 0 isn't legit, the next choice would be the direction 1, then 2 and so on. Following the **clockwise** order.



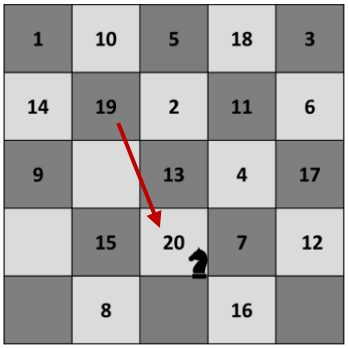
Example

Input chessboard already has 19 moves and the priority is set as 6.

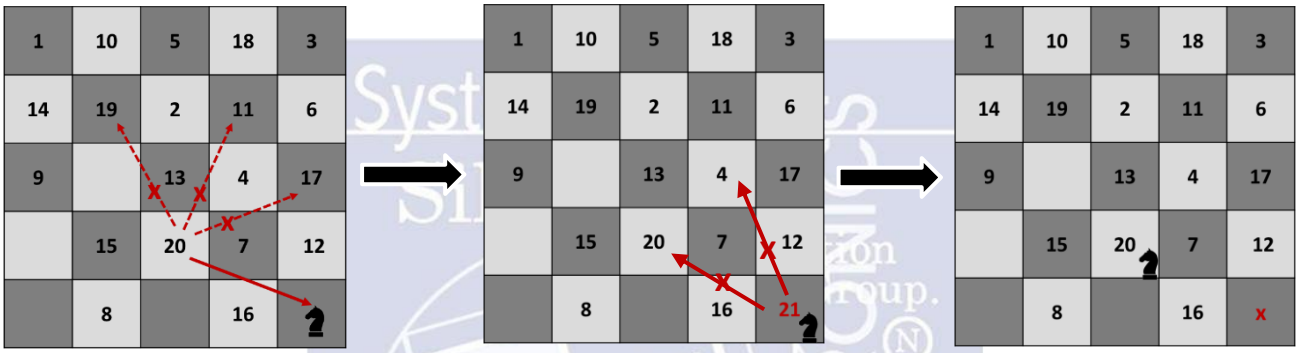
1	10	5	18	3
14	19	2	11	6
9		13	4	17
	15		7	12
	8		16	

move_num = 19
priority_num = 6

Choose direction 2, because direction 6, 7, 0, 1 aren't legit.



Due to the direction 6, 7, 0 are not legit, choose the direction 1 and move to (4,4). However, the knight has nowhere to move. It needs to move back to the previous step, which means step 20 (2,2).

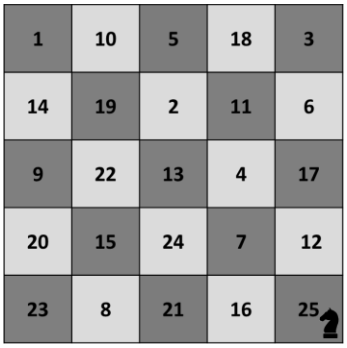


Following the rules, the knight can finally visit all squares. Then output the coordinate that the knight visited from step 1 to 25. Also output the sequence of order simultaneously.

- In this example：

The coordinate outputs are: (0,0)→ (1,2) → (0,4) → …… → (3,2) → (4,4).

The sequence of order outputs are : 1→ 2 → 3……→ 25



Input Signal	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when input signals are valid.
in_x	3	The move's x coordinate.
in_y	3	The move's y coordinate.
move_num	5	The number of moves that already be done in input chessboard.
priority_num	3	The priority of knight's moving strategy for each step.

Output Signal	Bit Width	Definition
out_valid	1	High when output is valid.
move_out	5	The order of the move.
out_x	3	The move's x coordinate.
out_y	3	The move's y coordinate.

Inputs

1. You will receive **move_num[4:0]** and **priority_num[2:0]** in the first cycle when **in_valid** is high.
2. The **in_x[2:0]** and **in_y[2:0]** might be valid 1 ~ 24 cycles which defined by **move_num[4:0]** and the order of input is the order of knight's moves.
3. All input signals will be synchronized at **negative edge** of the clock.
4. The next input pattern will be triggered 1 ~ 5 cycles after **out_valid** falls.
5. Each pattern **only has one solution**.

Outputs

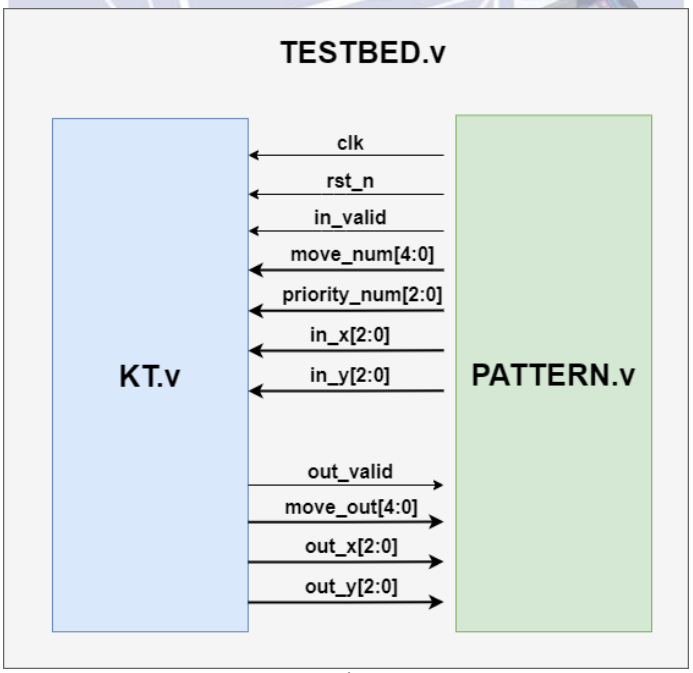
1. All outputs should be low after **rst_n** assert
2. **out_valid** should not be raised when **in_valid** is high.
3. Output signals should be synchronized at clock positive edge.
4. **out_valid** is set to high when the output value is valid and it will be high for 25 cycles continuously when triggered.
5. TA's pattern will capture your output for checking at **negative** clock edge.

Specifications

1. Top module name：KT (Filename: KT.v)
2. It is 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. The clock period of the design is **5ns.**

4. The **rst_n** would be triggered only once. All output signals should be reset to low after the reset signal is asserted.
5. The out_valid and output data must be asserted successively **25** cycles.
6. The next group of inputs will come in 1~5 cycles after your out_valid pull down.
7. The synthesis result of data type cannot include any LATCH.
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 10000 cycles.
11. The look-up table method is forbidden. (TA will check your design)
12. 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.
13. Don't write chinese comments or other language comments in the file you turned in.
14. 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.
15. 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.
16. 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: 30%

   - Area * Latency time 30%

```
                    Congratulations!
            You have passed all patterns!
            Your execution cycles = 48786 cycles
            Your clock period = 5.0 ns
            Your total latency = 243930.0 ns
```

2. **Please upload the following file on new e3 platform before 12:00 p.m. on 10/4:**
   - KT_iclabxx.v (xx is your account number, i.e. KT_iclab99.v)
   - **If the uploaded files 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 the design if there's latch or not in ***syn.log***)

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

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

Sample Waveform

1. 20 cycles for input knight's moves

2. asynchronous reset and active-low and reset all output

3. 25 cycles for output order of knight's moves