NCTU-EE DCS – 2017

Lab05 Exercise

Deadline:4/13(Thursday) 10:00 a.m.

Design: FSM (Sorting and Searching)

Data Preparation

1. Extract test data from TA’s directory:

   % tar xvf ~ dcsta02/LAB05.tar

2. The extracted LAB director contains:

   a. EXERCISE/ : Exercise

Design Description

In this lab, we wish you design a simple finite-state-machine (FSM). The FSM in the sequential circuits is necessary when you need to solve the complex problems.

Sorting and searching are two common algorithms in data structure. In some applications, sorting can be used to easily know data information and searching can help us to find specific data which we interested and know the relative location. There are many sorting algorithm such as bubble sort, quick sort, insertion sort and so on while searching such as depth-first search, breadth-first search and so on.

In this design, you will get some integer value, and do the sorting from large to small. After the sorting, you need to know the specific value location. If there are values repeated, the order ahead will be chosen. You will receive 7 values and one searching value in the input.

EX1:

The input value is 5, 10, 15, 20, 25, 30, 35. The searching value is 20.

You should output value 35, 30, 25, 20, 15, 10, 5. And the order is 4.

EX2:

The input sequence is 10, 50, 40, 80, 30, 30, 100. The searching value is 30.

You should output value 100, 80, 50, 40, 30, 30, 10. And the order is 5.

We recommend you can use the following states to help you finish the practice. You may need counters in your design also.

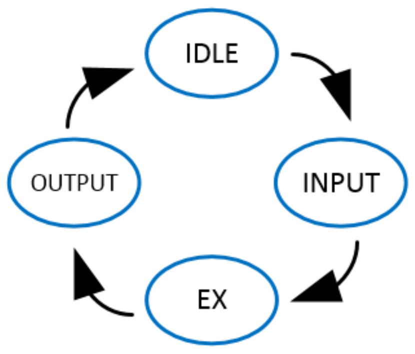
State list:

a. IDLE

b. INPUT

c. EX

d. OUTPUT



Inputs

1. All input signals will be **synchronized** at **negative edge** of the clock.

I/O	Signal name	Description
Input	**clk**	Clock
Input	**rst_n**	Asynchronous active-low reset. You should set all your outputs to 0 when rst_n is low
Input	**in_valid**	in[6:0] and in_s[6:0] are valid when in_valid is high.
Input	**in[6:0]**	The unsigned **in[6:0]** is valid only when **in_valid** is high, and is delivered 7 cycles continuously.
Input	**in_s[6:0]**	The unsigned **in_s[6:0]** is the searching value and is delivered in first cycle when **in_valid** is high.

Outputs

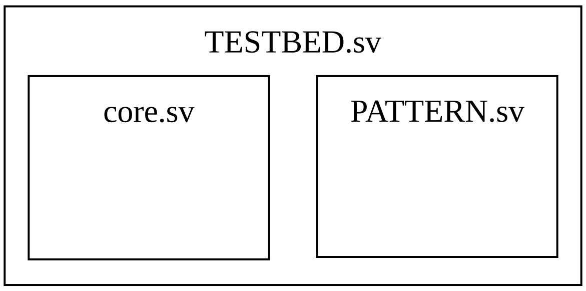
1. Output signals are synchronized at clock positive edge.
2. The TA's pattern will capture your output for checking at clock negative edge.

I/O	Signal name	Description
Output	**out[6:0]**	**out** should be printed in descending order in continuous **7** cycles.
Output	**order[2:0]**	**order** should be printed in the first cycle only when **out_valid** is high
Output	**out_valid**	**out_valid** should be low after initial reset and not be raised when **in_valid is high**. **out_valid** should be set to high when your output value is ready, and will be high for **7** cycles continuously.

Specifications

1. Top module name ： **core**(Filename: **core.sv**)
2. Input ports: **clk, rst_n, in_valid, in[6:0], in_s[6:0]**
3. Output ports: **out[6:0], order[2:0], out_valid**
4. It is an **asynchronous active-low reset** and **positive edge clk** architecture
5. The clock period of the design is **3ns**.
6. The input delay is set to **1.5ns**.
7. The output delay is set to **1.5ns**, and the output loading is set to **0.05**.
8. The **out_valid** and output value must be asserted successively 7 cycles
9. The next group of inputs will come in **1~9** cycles after your **out_valid** is over.
10. After synthesis, please check **syn.log** file that can not include any **Latch & Error**.
11. After synthesis, you can check CORE.area and CORE.timing. The area report is valid when the slack in the end of **timing report** is **non-negative**.
12. The latency for one case should be smaller than **1000 cycles**.

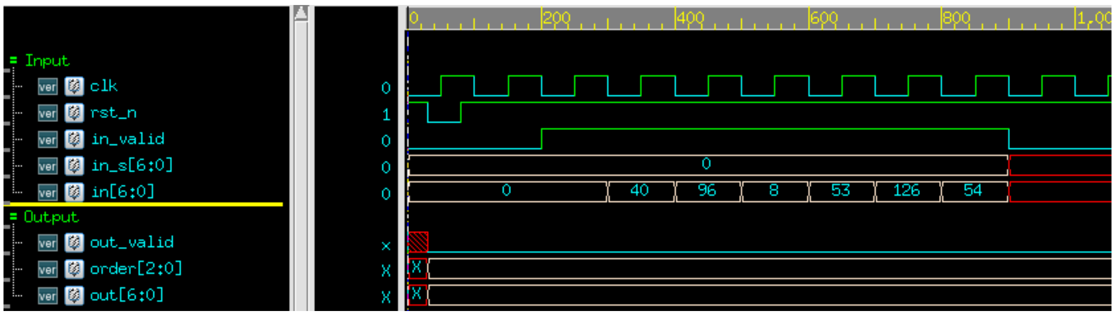
Block Diagram

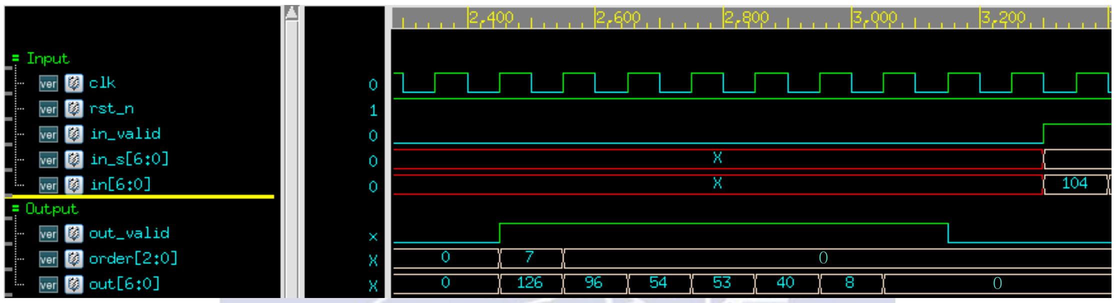


Note

1. Grading policy:
   a. Pass the RTL simulation
   b. Synthesis successful
   (**no Latch and Error**)
   (**slack in the timing report is non-negative**)
   c. Using For loop is not allowed.
   d. Plagiarism is not allowed.
   e. Function correct 80% + Area 20%
2. Template folders and reference commands:
   a. 01_RTL/ (RTL simulation) **./01_run**
   b. 02_SYN/ (Synthesis) **./01_run_dc**
   (Check the design which contain **latch** or not in **syn.log**
   (Check the design's timing in /Report/**core.timing**)

Sample Waveform


Input
clk
rst_n
in_valid
in_s[6:0]
in[6:0]
Output
out_valid
order[2:0]
out[6:0]
0
200
400
600
800
0
40
96
8
53
126
54


Input
clk
rst_n
in_valid
in_s[6:0]
in[6:0]
Output
out_valid
order[2:0]
out[6:0]
2,400
2,600
2,800
3,000
3,200
104
7
126
96
54
53
40
8