2023 Digital IC Design

Midterm Exam

Meeting Room Number：2

Please check if the last digit of your student id is equal to the meeting room number. Each meeting room has different exam questions. Tas will verify your code according to your student id, the mismatch between student id and meeting room number may cause you to lose part of the score.

Question 1: Bitonic Sorter

A bitonic sequence is a sequence of numbers that is constructed with an ascending subsequence and a descending subsequence. Bitonic sorter is a combinational digital circuit that sorts a sequence of numbers by constructing and merging bitonic sequences. In this question, you are requested to design an 8-number bitonic sorter, which arranges 8 input numbers in **descending order**. The bitonic sorter could be constructed by two type of swapper, Ascending Swapper (AS) and Descending Swapper (DS). The specifications and functions of the AS, DS and bitonic sorter are detailed in the following sections.

1.1. Ascending Swapper:

An ascending swapper will rearrange the input values to make sure the output values are in ascending order. That is, ***number_out1*** will be assigned the smaller value between ***number_in1*** and ***number_in2***, and ***number_out2*** will be assigned the larger value between ***number_in1*** and ***number_in2***.

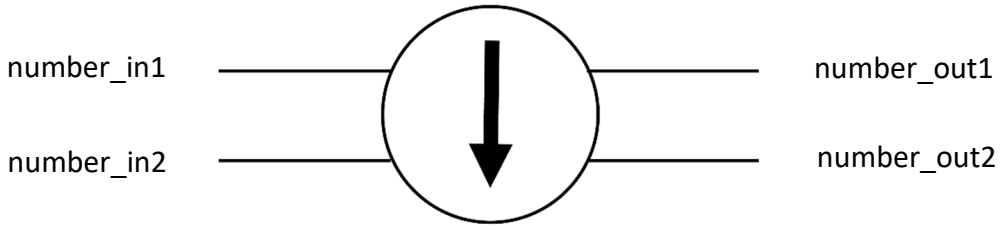

Fig. 1. An ascending swapper.

1.2. Descending Swapper:

A descending swapper will rearrange the input values to make sure the output values are in descending order. That is, ***number_out1*** will be assigned the larger value between ***number_in1*** and ***number_in2***, and ***number_out2*** will be assigned the smaller value between ***number_in1*** and ***number_in2***.

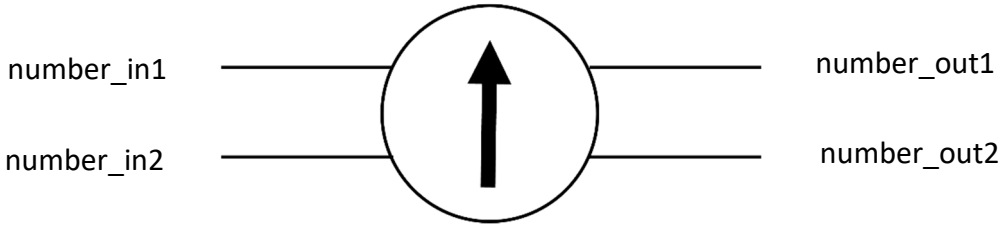

Fig. 2. A descending swapper.

Table I. I/O interface of the AS and DS module

Signal Name	IO	width
number_in1	I	8
number_in2	I	8
number_out1	O	8
number_out2	O	8

1.3 Bitonic Sorter

An 8-number Bitonic Sorter can be constructed with several AS and DS modules. The architecture of an 8-number Bitonic Sorter is illustrated in Fig. 3. The arrows directing from top to bottom denote ascending swappers, while the arrows directing from bottom to top denote descending swappers.

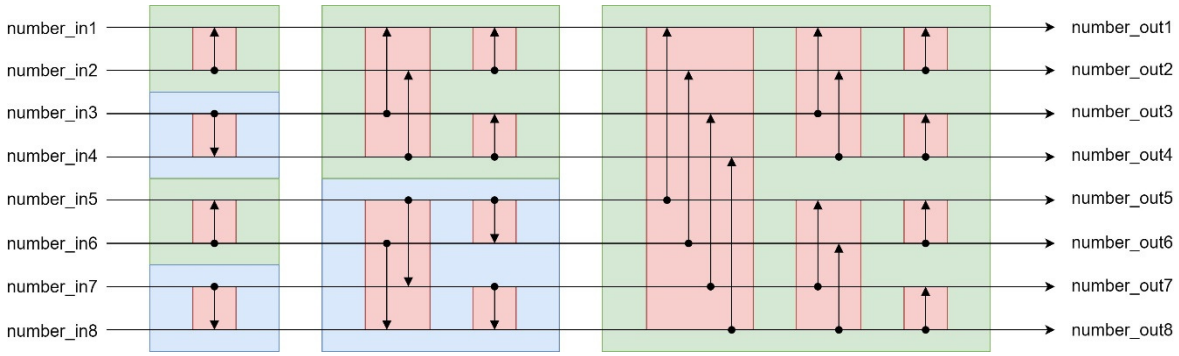

Fig. 3. The architecture of 8-number bitonic sorter.

For ease of scoring, the bitonic sorter architecture is divided into three separate modules. The three modules respectively correspond to stage1(S1), stage2(S2), and stage3(S3) of the sorting process, which is shown in Fig. 4. In S1, the input numbers are rearranged to construct two bitonic sequences. In S2, the two bitonic sequences are merged into two sorted subsequences. One of them is in ascending order, while the other one is in descending order. The two subsequences can also be viewed as a bitonic sequence. Finally, the bitonic sequence of 8 numbers is rearranged to a sequence in ascending order in S3.

Please design the three modules of the bitonic sorter with your AS and DS modules. Otherwise, no score will be given. The testbench will call the three modules to validate the function of the circuit. **Notably, there are dependencies between the three modules. To pass the test patterns, the output from the previous stage must be correct.**

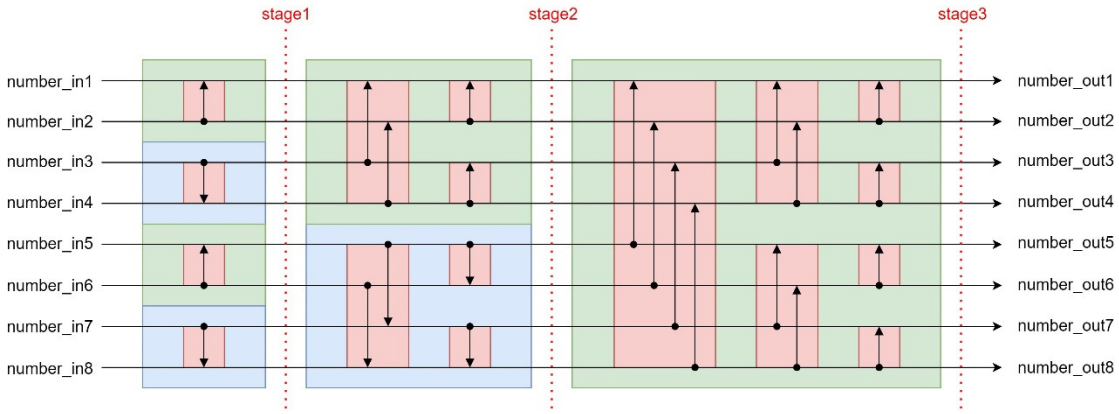

Fig. 4. The three stages of bitonic sorting.

Table II. I/O interface of the modules of S1, S2, and S3

Signal Name	IO	Width	Signal Name	IO	Width
number_in1	I	8	number_in5	I	8
number_in2	I	8	number_in6	I	8
number_in3	I	8	number_in7	I	8
number_in4	I	8	number_in8	I	8
number_out1	O	8	number_out5	O	8
number_out2	O	8	number_ou6	O	8
number_out3	O	8	number_out7	O	8
number_out4	O	8	number_out8	O	8

1.4 File Description:

File Name	Description
BITONIC_AS.v	The module of ascending swapper (AS).
BITONIC_DS.v	The module of descending swapper (DS).
BITONIC_S1.v	The module of bitonic sorter stage 1
BITONIC_S2.v	The module of bitonic sorter stage 2
BITONIC_S3.v	The module of bitonic sorter stage 3
BITONIC_tb.sv	The testbench file. The content in this file is **not allowed** to be modified.
sw_test_data.dat	Test data for AS and DS verification.

test_data.dat	Test data for 8-number bitonic sorter verification.
as_golden_data.dat	Golden data for AS verification.
ds_golden_data.dat	Golden data for DS verification.
s1_golden_data.dat	Golden data for bitonic sorter stage1 verification.
s2_golden_data.dat	Golden data for bitonic sorter stage2 verification.
s3_golden_data.dat	Golden data for bitonic sorter stage3 verification.

1.5 Scoring of Question 1 [50%]

The scoring is fully based on the functional simulation results in Modelsim. There is no necessary to synthesis the Verilog codes.

1.5.1: Ascending Swapper and Descending Swapper [10%]

All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

```
# --------------- Simulation Start ----------------
#
# --          Acending Swapper PASS!           --
# --          Descending Swapper PASS!         --
# -------------------------------------------------
```

Fig. 5. Simulation result for DS,AS module.

1.5.2 Stage 1 of bitonic sorting [10%]

All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

```
# -------------------------------------------------
# --          Stage 1 Simulation finish        --
# --                 Stage 1 PASS!             --
# -------------------------------------------------
```

Fig. 6. Simulation result for Bitonic_S1 module.

1.5.3 Stage 2 of bitonic sorting [10%]

All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

```
# -------------------------------------------------
# --          Stage 2 Simulation finish        --
# --                 Stage 2 PASS!             --
# -------------------------------------------------
```

Fig. 7. Simulation result for Bitonic_S2 module.

1.5.4 Stage 3 of bitonic sorting [20%]

All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

```
# -------------------------------------------------
# --          Stage 3 Simulation finish        --
# --                 Stage 3 PASS!             --
```

Fig. 8. Simulation result for Bitonic_S3 module.

Question 2: Rails

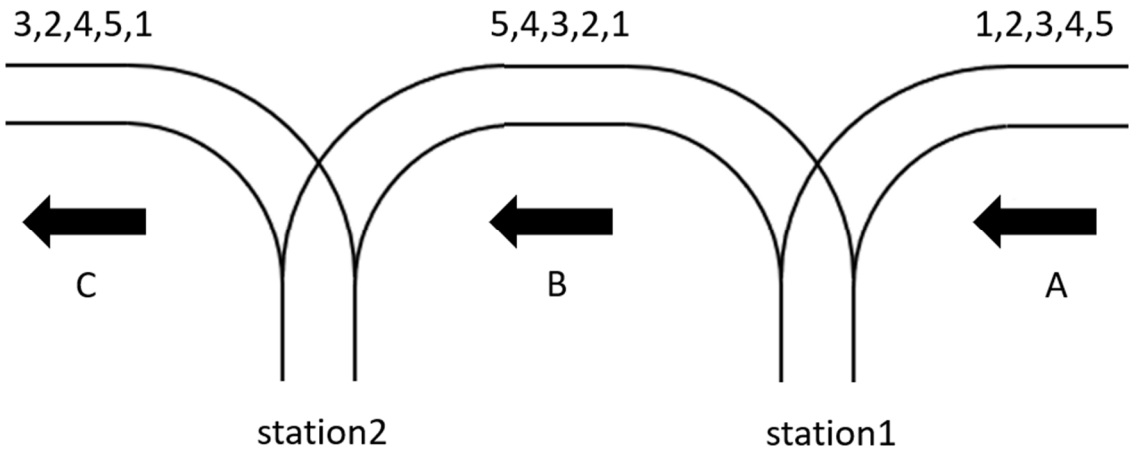

Fig. 9. Rails.

There are two railway stations that established only a surface track. Furthermore, the stations are designed as dead-end stations, as shown in Fig. 9. In this question, you are requested to design a circuit that determines whether it is possible to meet the given departure order. The specification and function of the circuit is detailed in the following sections

2.1 Block Overview

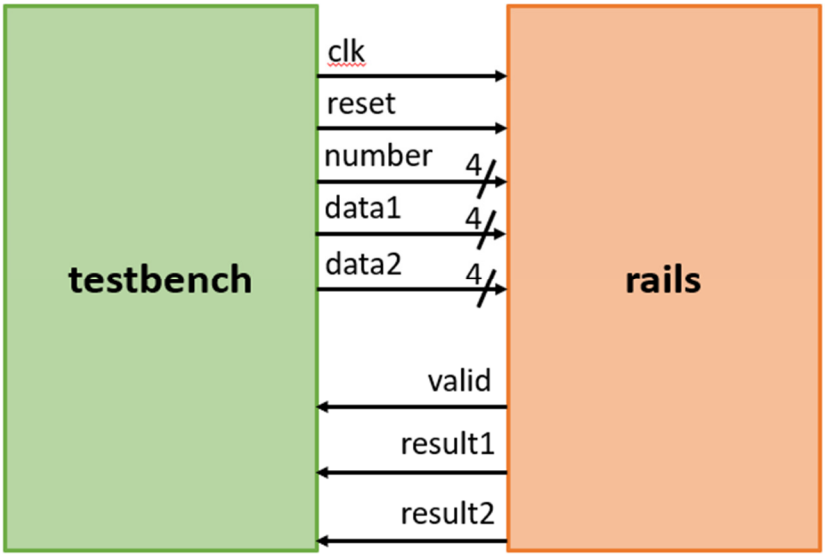

Fig. 10. The block overview.

2.2 I/O Interface

Table III. I/O interface of the rails module.

Signal Name	I/O	width	Description
clk	I	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
reset	I	1	Active-high asynchronous reset signal.
number	I	4	The number of trains
data1	I	4	The required departure order leaving station1.
data2	I	4	The required departure order leaving station2.
valid	O	1	When output results, set *valid* signal to high. The testbench will check the output results when *valid* signal is high.
result1	O	1	If the given departure order leaving station1 is possible to meet, *result1* should be set as high. Otherwise, it has to be set as low.
result2	O	1	If the given departure order leaving station2 is possible to meet, *result2* should be set as high. Otherwise, it has to be set as low.

2.3 File Description

File Name	Description
rails.v	The module of rails, which is the top module in this design.
tb.v	The testbench file. The content in this file is **not allowed** to be modified.
rails_num.dat	Test data for rails verification.
test_data_rails1.dat	Test data for rails verification.
test_data_rails2.dat	Test data for rails verification.
golden_data_rails.dat	Golden data for rails verification.

2.4 Function Description

2.4.1 Description

Every train arrives at station 1 from direction A, and departs station 1 from direction B. The trains departing from station 1 will enter station 2 from direction B, and leave station 2 from direction C. The trains will be numbered in ascending order from 1 to N according to the order in which they arrived at station 1, and the number of coming trains ranges from 3 to 10. Given two sequences of departure

order, the circuit has to determine if the trains can leave station 1 and station 2 in the required order. In the station, the order of trains can't be changed. For any train, there is no limit to its arrival time and staying time at the station. However, once a train has entered the station, it cannot return to the track of its coming direction. After a train has left the station, it cannot return to the station, either. Notably, the arriving order of trains to station 2 is the departure order from station 1. To simplify the question, the departure order leaving station 2 is regarded as impossible to meet if the departure order leaving station 1 can't be met. That is, *result2* must be false if *result1* is false.

In this question, there is capacity limits for both train stations. The maximum number of trains staying in station 1 is 7, and the maximum number of trains staying in station 2 is 5. If the number of trains in the station has reached the limitation, arriving trains can't enter the station until at least one train leaves the station.

Hint: The stations can be considered as two 'stacks'. When the number of elements in the stack reaches the limitation, only 'pop' operation is allowed.

2.4.2 Timing Specifications

After the system resets, the ***number*** signal will first input the number of coming trains for the first test pattern (T1), followed by the required departure order in sequence at each cycle (T2~T6). Once the testbench has finished inputting the input pattern, it will wait for the rails circuit to output a result. When the results are ready to be outputted, the ***valid*** signal must be set as high. The signals ***result1*** and ***result2*** represent if the departure order leaving station 1 and station 2 can be met, respectively. The testbench will validate the results when it detects a high-level ***valid*** signal (T7). At the next cycle, ***valid*** should be pulled down as low to avoid the testbench from misjudgment (T8). After ***valid*** is pulled down, the testbench will start inputting the next test pattern from the next cycle (T9).

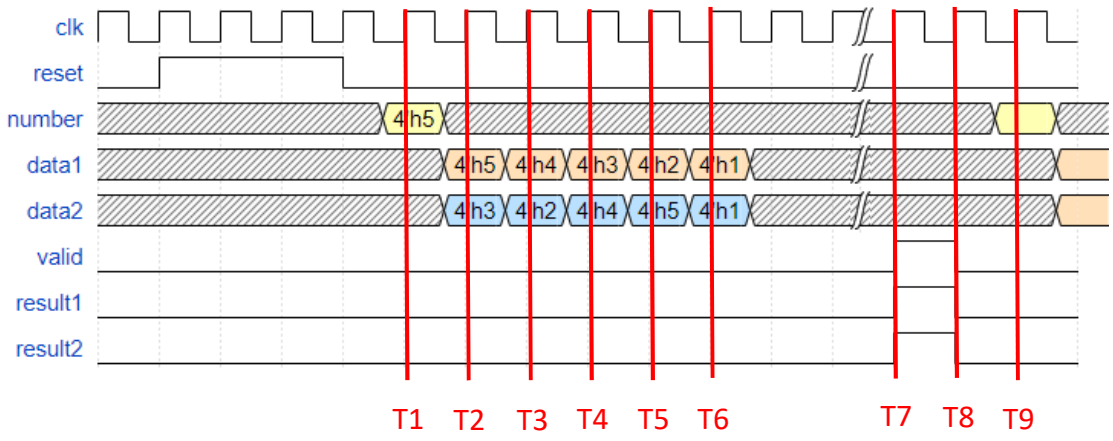

Fig. 11. Timing diagram of input and output.

2.5 Scoring of Question 2 [50%]

The scoring is fully based on the functional simulation results in Modelsim. There is no necessary to synthesis the Verilog codes. Please don't design specifically for the test pattern. Otherwise, you will get 0 point.

There are three test patterns in this question, testbench will verify if the outputs are correctly generated. If all the results are correct, you will get the following message in ModelSim simulation.

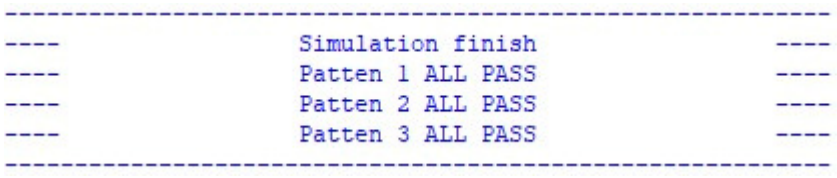

Fig. 12. Simulation result for all passes.

The scoring of the three test patterns are as following：

- **Pattern 1 [10%]**
- **Pattern 2 [20%]**
- **Pattern 3 [20%]**

Submission:

1. Submitted files

You should classify your files into two directories and compress them to .zip format. The naming rule is Mid_studentID_name.zip. If your file is not named according to the naming rule, you will lose five points. Please submit your .zip file to folder Mid in moodle.

	Mid_studentID_name.zip
Q1	
*.v	All of your Verilog RTL code for Question 1
Q2	
*.v	All of your Verilog RTL code for Question 2

2. Notes

Please do not modify any content of testfixture.

Deadline: 2023/4/18 12:00.

No late submissions will be accepted, please pay attention to the deadline.