2024 Digital IC Design
Final Exam
Meeting Room Number:5

Please check your meeting room number is correct which is announced at moodle. 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: Two Matrix Multiplication with illegal input/output

This assignment aims for you to create a matrix multiplication module which can do two matrices multiplication and can detect illegal input such as illegal matrix or matrices that could not be multiplicated. It can also detect illegal output such as overflow. The specification and function of the circuit is detailed in the following sections.

1. Design Specifications

1.1 I/O Interface

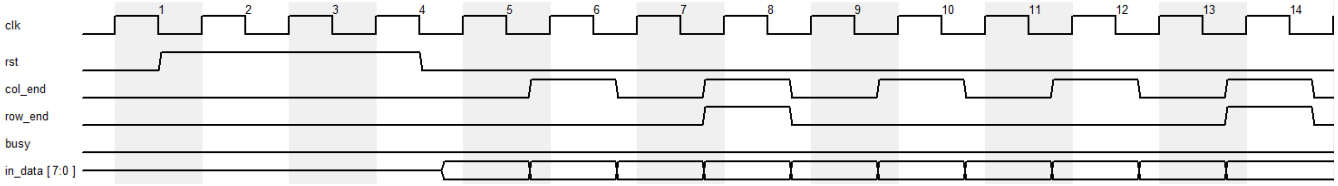
Table I. I/O interface of the design

Signal Name	I/O	width	Description
Clk	I	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
Rst	I	1	Active-high asynchronous reset signal.
in_data	I	8	Input 8 bit signed matrix data one by one.
col_end	I	1	This signal is 1 when the input matrix completes the data input of a row, otherwise it is 0.
row_end	I	1	This signal is 1 when an input matrix complete data input, otherwise it is 0.
valid	O	1	When this signal is 1, tb will check out_data, is_legal,ep,overflow and change_row signals.
out_data	O	12	Output the 12-bit matrix operation result, one element at a time.
is_legal	O	1	Output whether two matrices are legal and can be multiplied, 1 means they are legal and can be multiplied.
change_row	O	1	This signal is 1, When the output matrix completes the output of one row
ep	O	2	2-bit matrix legal signal, if matrix1 is legal and matrix2 is illegal, output ep = 2'b10, if all are

			legal, output ep = 2'b00, if all are illegal, output ep = 2'b11
busy	O	1	This signal is 1 means the system is busy and tb will stop in_data transmission.
overflow	O	1	This signal is 1 when the result matrix element overflow, otherwise it is 0.

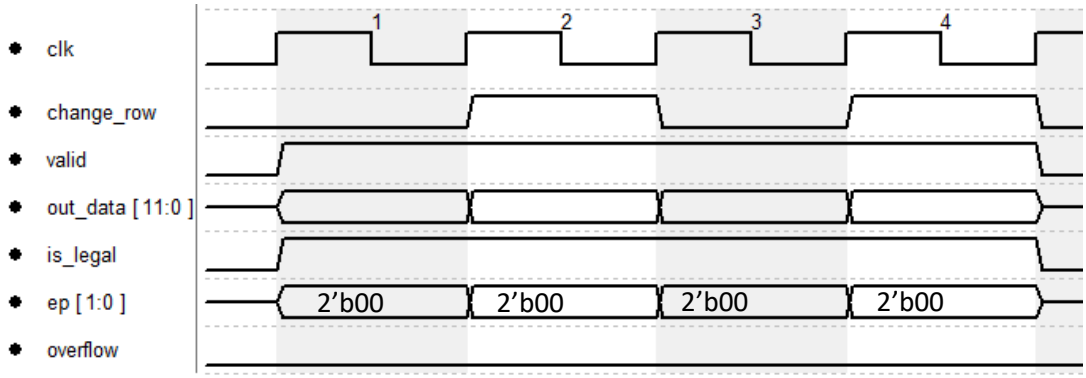
1.2 System Description

1. After the system is reset, when tb detects that busy is 0, the matrix data is input one by one from in_data.
2. The shape of the input matrices is controlled by row_end and col_end signal. The following is an example of inputting a 2 x 2 matrix and a 3 x 2 matrix:



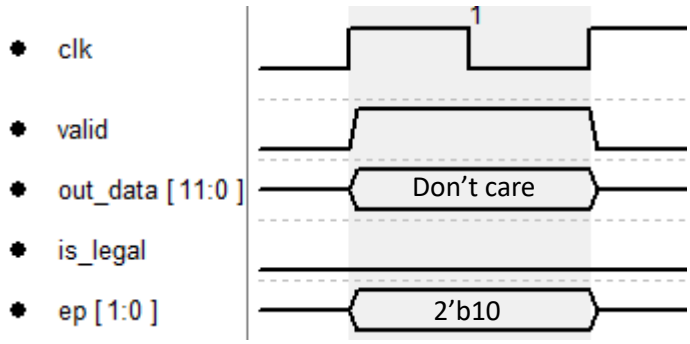
Input Timing Diagram

3. The shape of the output matrices is controlled by change_row signal. The following is an example of outputting a 2 x 2 matrix:



Output Timing Diagram

4. The following is an example of outputting the result of legal matrix1,illegal matrix2:

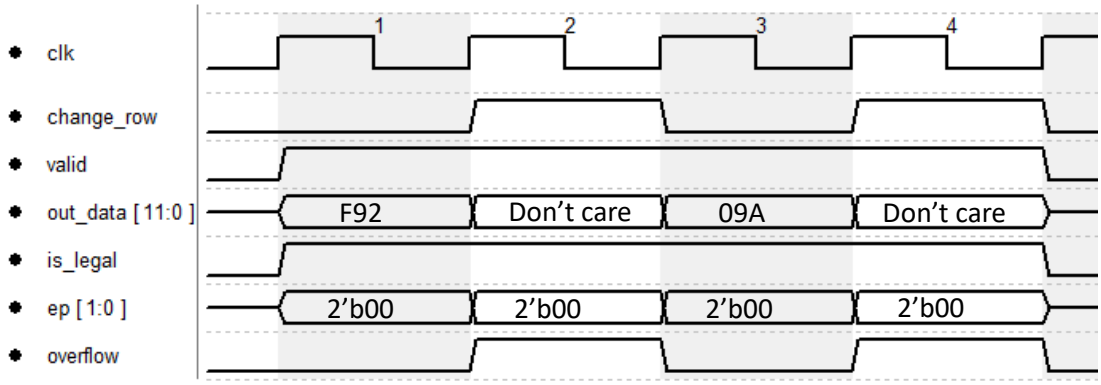


5. Illegal matrix example:

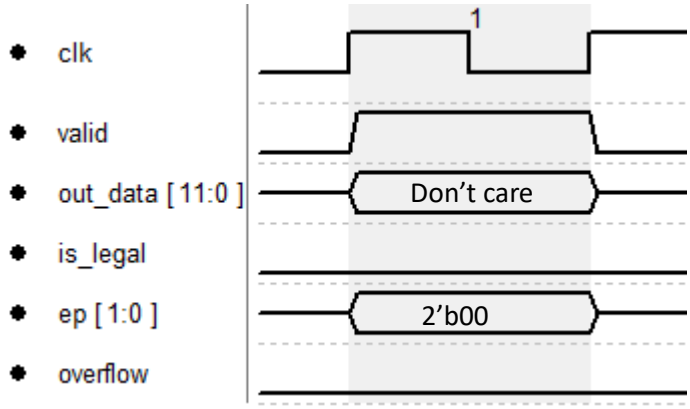
$$\begin{bmatrix} 0C & \\ 14 & F9 \end{bmatrix}$$

6. The following is an example of outputting a overflow matrix:

$$\begin{bmatrix} F92 & overflow \\ 09A & overflow \end{bmatrix}$$



7. The following is an example of outputting the result of 2 legal matrix but can not multiply:



Notice:

1. When the second matrix is completely input and valid is 1, tb will check whether out_data, is_legal, change_row,overflow and ep are correct.(Please keep valid 0 until the second matrix is completely input and output calculation is finished)

2. If any matrix is illegal, only ep and is_legal will be compared; if the matrices are all legal but cannot be multiplied, only ep and is_legal will be compared.

3. If 4 pieces of data are to be output in this round, but only 3 are output, tb will continue to wait for the output of the fourth data and will not give the next matrix data.

4. Please read the data in the same matrix continuously and do not read it in sections.

2. File Description

File Name	Description
MM.v	The top module of your design.
testfixture.v	The testbench file.
in1.dat in2.dat in3.dat	Input data
out1.dat out2.dat out3.dat	Golden data
ep1.dat ep2.dat ep3.dat	ep data
mx_shape1.dat mx_shape2.dat mx_shape3.dat	Matrix shape data
overflow1.dat overflow2.dat overflow3.dat	Overflow data

3. Scoring of Question 1 [40%]

The scoring is fully based on the functional simulation results in Modelsim. There is no necessary to synthesis the Verilog codes. All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

(You won't receive partial credit for partially correct answers.)

Pattern1:all legal matrix,no overflow case(10%)

```
#  6:Pattern         181 is PASS !
# ---------------------------- Simulation FINISH !!----------------------------
# ============================================================================
#
#  \(^o^)/ CONGRATULATIONS!!  The simulation result is PASS!!!
#
# ============================================================================
# ** Note: $stop    : C:/Users/user/Desktop/finalv2/testfixture.v(281)
```

Pattern2:contain illegal matrix,no overflow case(15%)

```
#  1:Pattern         279 is PASS !
# ---------------------------- Simulation FINISH !!----------------------------
# ================================================================================
#
#  \(^o^)/ CONGRATULATIONS!!  The simulation result is PASS!!!
#
# ================================================================================
# ** Note: $stop    : C:/Users/user/Desktop/finalv2/testfixture.v(281)
```

Pattern3:all legal matrix with overflow case(15%)

```
 1:Pattern         421 is PASS !
---------------------------- Simulation FINISH !!----------------------------
================================================================================

 \(^o^)/ CONGRATULATIONS!!  The simulation result is PASS!!!

================================================================================
** Note: $stop    : C:/Users/user/Desktop/finalv2/testfixture.v(281)
```

Question 2: Three Matrix Multiplication with illegal input

This assignment aims for you to create a matrix multiplication module which can do Three matrices multiplication and can detect illegal input such as illegal matrix or matrices that could not be multiplicated. The specification and function of the circuit is detailed in the following sections.

1. Design Specifications

1.1 I/O Interface

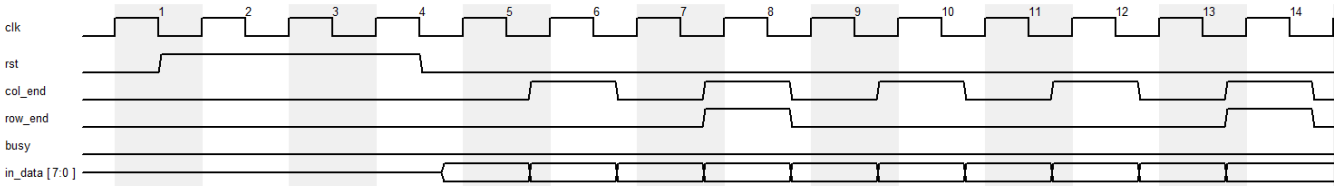
Table I. I/O interface of the design

Signal Name	I/O	width	Description
clk	I	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
rst	I	1	Active-high asynchronous reset signal.
in_data	I	8	Input 8 bit signed matrix data one by one.
col_end	I	1	This signal is 1 when the input matrix completes the data input of a row, otherwise it is 0.
row_end	I	1	This signal is 1 when an input matrix complete data input, otherwise it is 0.
valid	O	1	When this signal is 1, tb will check out_data, is_legal,ep and change_row signals.
out_data	O	32	Output the 32-bit matrix operation result, one element at a time.
is_legal	O	1	Output whether three matrices are legal and can be

			multiplied, 1 means they are legal and can be multiplied.
change_row	O	1	This signal is 1, When the output matrix completes the output of one row
ep	O	3	3-bit matrix legal signal, if matrix1 is legal, matrix2 and matrix3 is illegal, output ep = 3'b110, if all are legal, output ep = 3'b000, if all are illegal, output ep = 3'b111
busy	O	1	This signal is 1 means the system is busy and tb will stop in_data transmission.

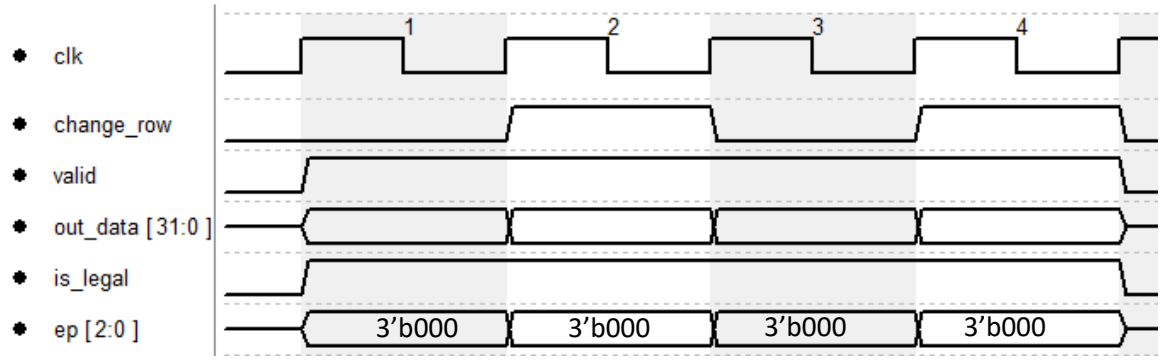
1.2 System Description

1. After the system is reset, when tb detects that busy is 0, the matrix data is input one by one from in_data.
2. The shape of the input matrices is controlled by row_end and col_end signal. The following is an example of inputting a 2 x 2 matrix and a 3 x 2 matrix:



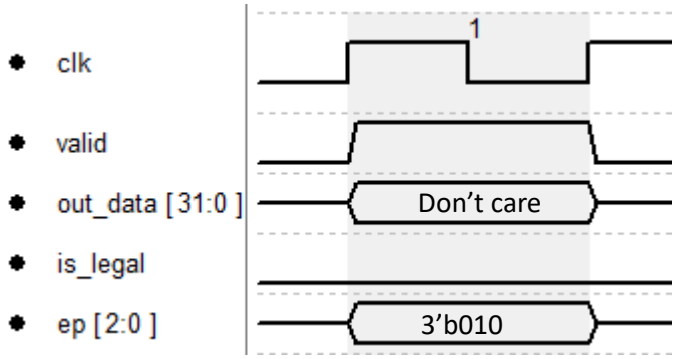
Input Timing Diagram

3. The shape of the output matrices is controlled by change_row signal. The following is an example of outputting a 2 x 2 matrix:

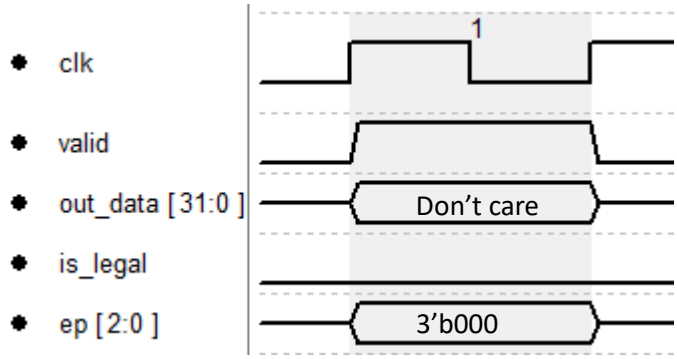


Output Timing Diagram

4. The following is an example of outputting the result of legal matrix1,illegal matrix2 and legal matrix3:



5. The following is an example of outputting the result of 3 legal matrix but can not multiply:



Notice:

1. When the third matrix is completely input and valid is 1, tb will check whether out_data, is_legal, change_row and ep are correct.(Please keep valid 0 until the third matrix is completely input and output calculation is finished.)
2. If any matrix is illegal, only ep and is_legal will be compared; if the matrices are all
3. legal but cannot be multiplied, only ep and is_legal will be compared.
4. If 4 pieces of data are to be output in this round, but only 3 are output, tb will continue to wait for the output of the fourth data and will not give the next matrix data.
5. Please read the data in the same matrix continuously and do not read it in sections.

2. File Description

File Name	Description
MM.v	The top module of your design.
testfixture.v	The testbench file.
in.dat	Input data
out.dat	Golden data

mx_shape.dat	Matrix shape data

3. Scoring of Question 2 [10%]

The scoring is fully based on the functional simulation results in Modelsim. There is no necessary to synthesis the Verilog codes. All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

(You won't receive partial credit for partially correct answers.)

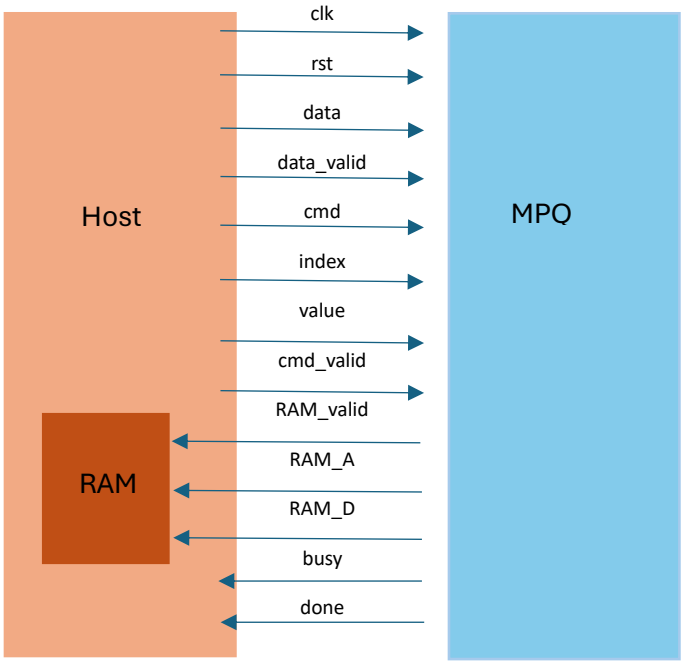
```
#  1:Pattern         272 is PASS !
# ---------------------------- Simulation FINISH !!----------------------------
# ============================================================================
#
#  \(^o^)/ CONGRATULATIONS!!  The simulation result is PASS!!!
#
# ============================================================================
# ** Note: $stop    : C:/Users/user/Desktop/Q2/testfixture.v(292)
```

Question 3: Max-Priority Queue

This assignment aims for you to complete the circuit design of the Max-Priority Queue. This circuit can read data and performing the functions of Build_Queue, Extract_Max, Increase_Value, and Insert_Data according to specified control commands. Finally, the results will be written to RAM.

1. Design Specifications

1.1 Block Overview:



1.2 I/O Interface:

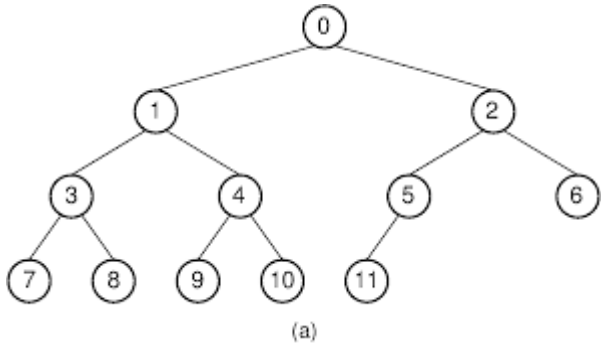
Table I. I/O interface of the design

Signal	I/O	Bit Width	Description
clk	input	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
rst	input	1	Active-high asynchronous reset signal.
data	input	8	Original Data: When "data_valid" is high, it indicates that the data is valid.

data_valid	Input	1	When this signal is high, it indicates that the data input is valid.
cmd	input	3	Command Input Signals: There are five types of command inputs for this controller. Command inputs are valid only when "cmd_valid" is high and "busy" is low.
Index	input	8	Instruction-Related Signals, detailed below.
Value	input	8	Instruction-Related Signals, detailed below.
cmd_valid	input	1	When this signal is high, it indicates that the cmd instruction is a valid input command.
RAM_valid	output	1	RAM Memory Data Enable Signal: When it is high, it indicates that the data and address signals transmitted from the MPQ end are valid.
RAM_A	output	8	RAM Address Data: The MPQ end needs to use this output data to instruct the Host end's Image RAM memory to write data to the specified address.
RAM_D	output	8	RAM Data: The MPQ end utilizes this output signals to write data to the RAM memory module on the Host end.
busy	output	1	System Busy Signal: Description: When this signal is high, it indicates that the controller is executing the current command and cannot accept any new command inputs. When this signal is low, the system is ready to accept new commands. Upon reset, the default setting is high.
done	output	1	When the controller completes writing to RAM, setting "done" to high indicates completion.

1.3 RAM Correspondence and Timing Specifications

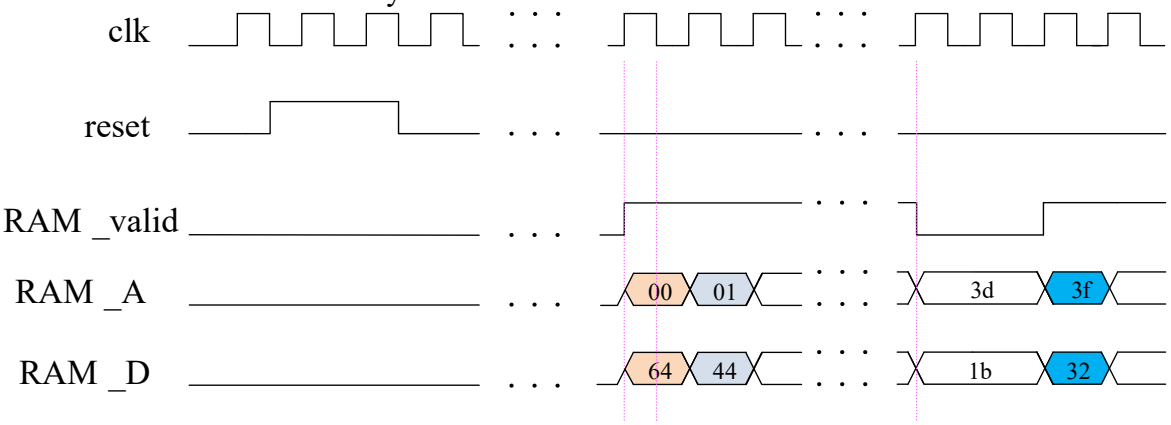
The result computed by the MPQ is a complete tree. The correspondence is as illustrated in the following diagram:



(a)

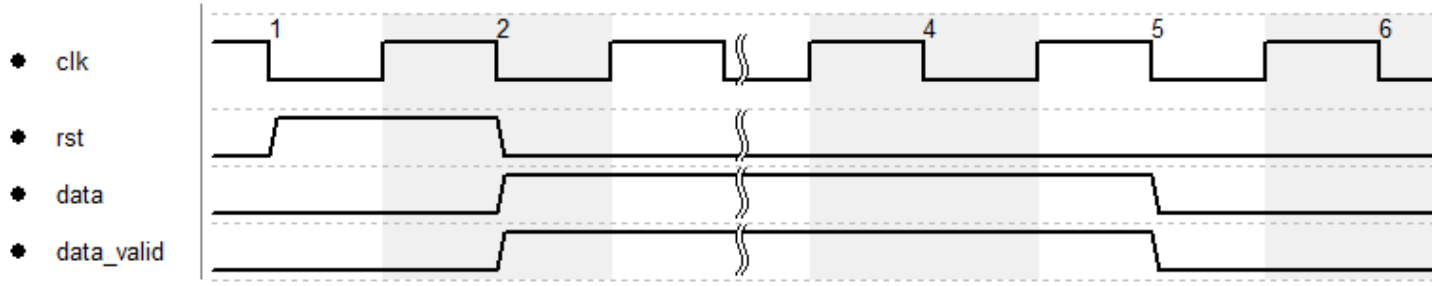
The timing of RAM output is as shown in Figure. If the RAM_valid signal is high (as indicated at time T1 in the Figure), the MPQ should simultaneously place the desired address on the RAM_A

output signals and the data on the RAM_D output signals at each positive edge of the clock signal. At the negative edge of the HOST clock signal at time T2, the system will perform a writing operation. When writing data, keep RAM_valid high and update RAM_A and RAM_D as necessary. To end the data write, set RAM_valid to low at time T3. This memory does not need to consider read and write latency.

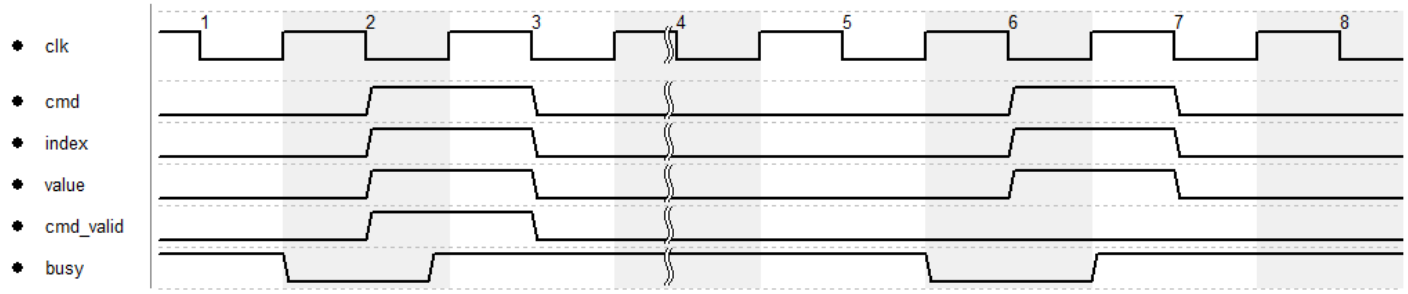


1.4 Timing Specification Diagram

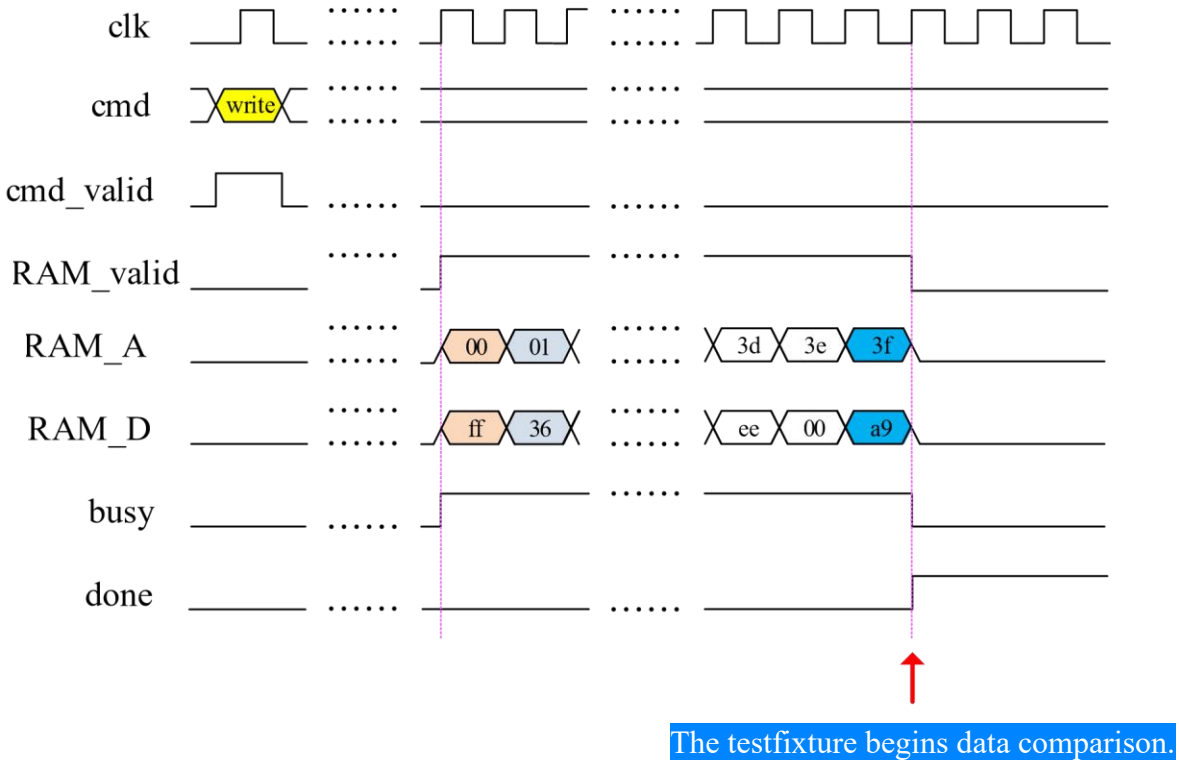
- The timing specification diagram after reset is shown in Figure.
  - After the circuit is reset, the controller will output n pieces of data.
  - During the entire process, the 'busy' signal stays high, showing it's not ready for commands.



- The timing specification diagram for control commands (Build_Queue, Extract_Max, Increase_Value, Insert_Data) is as shown in the following figure.
  - Throughout the entire processing, the "busy" signal remains high. After the operation is completed, "busy" is set back to low to accept new command inputs.



- The timing specification diagram for the write command is as shown in the following figure.
  - When executing the write command, the controller will write the processed image data into the RAM.
  - When RAM_valid is high, it indicates writing to RAM. At this point, address signals can be input to write image data into the RAM.
  - After writing is completed, the "done" signal should be set to high to indicate completion. At this point, the testfixture will compare the written data in RAM with the golden pattern.



Question 3(a): MIN-Priority Queue

This assignment aims for you to complete the circuit design of the Min-Priority Queue. This circuit is capable of reading data and performing the functions of Build_Queue, Extract_Min, Decrease_Value, and Insert_Data according to specified control commands. Finally, the results will be written to RAM. The specification and function of the circuit is detailed in the following sections.

1. File Description

File Name	Description
MQP.v	The module of MPQ, which is the top module in this design.
testfixture.v	The testbench file.
pat.dat	Input Raw Data: The data format is hexadecimal.
cmd.dat	Command Data: The data format is binary.
index.dat	Index Data: The data format is hexadecimal.
value.dat	Value Data: The data format is hexadecimal.
golden.dat	Golden data for MPQ verification.

2. Function Description

2.1 CMD description

Description	CMD	index	value
Build_Queue	0 (000)	0 (unused)	0 (unused)
Extract_Min	1 (001)	0 (unused)	0 (unused)
Decrease_Value	2 (010)	index	value
Insert_Data	3 (011)	0 (unused)	value
Write	4 (100)	0 (unused)	0 (unused)

Notice:

Build_Queue : Build min heap. You can modify HW4 reference code from build max heap to build min heap.

Extract_Min : Extract the root of MIN-Priority Queue. That is, return A[1] and remove A[1]. Then do heapify to maintain min heap if needed.

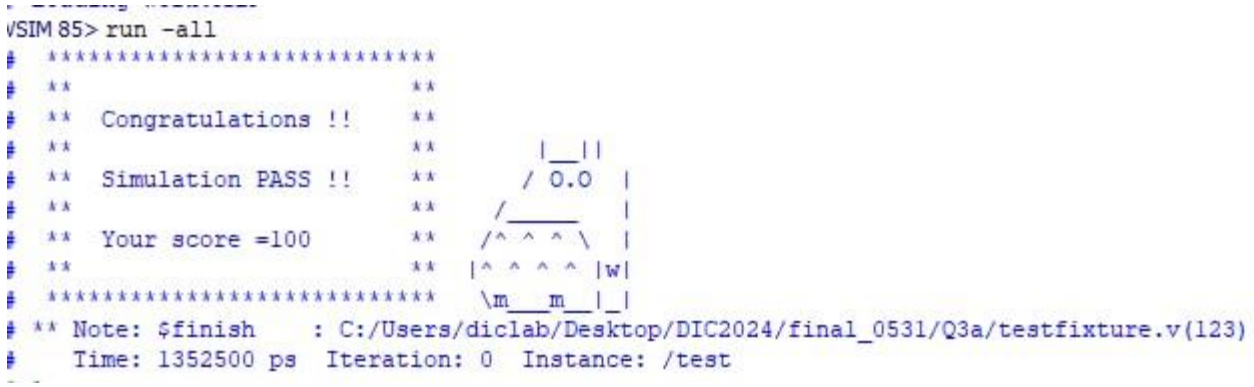
Decrease_Value : Set A[index] = value. Then do heapify to maintain min heap if needed. Notice that value will smaller than A[index]. That is, A[index] > value

Notice that A is stand for min heap array.

Scoring of Question 3(a) [20%]

The scoring is fully based on the functional simulation results in Modelsim. There is no necessary to synthesis the Verilog codes. All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

(You won't receive partial credit for partially correct answers.)



Question 3(b): Increase const command

This assignment aims for you to complete the circuit design of the Max-Priority Queue. This circuit can read data and performing the functions of Build_Queue, Extract_Max, Increase_Value, Insert_Data and Increase Const according to specified control commands. Finally, the results will be written to RAM. The specification and function of the circuit is detailed in the following sections.

1. File Description

File Name	Description
MQP.v	The module of MPQ, which is the top module in this design.
tb.v	The testbench file.
pat.dat	Input Raw Data: The data format is hexadecimal.
cmd.dat	Command Data: The data format is binary.
index.dat	Index Data: The data format is hexadecimal.
value.dat	Value Data: The data format is hexadecimal.
golden.dat	Golden data for MPQ verification.

2. Function Description

2.1 CMD description

Description	CMD	index	value
Build_Queue	0 (000)	0 (unused)	0 (unused)
Extract_Max	1 (001)	0 (unused)	0 (unused)
Increase_Value	2 (010)	index	value
Insert_Data	3 (011)	0 (unused)	value
Write	4 (100)	0 (unused)	0 (unused)
Increase_Const	5(101)	index	0(unused)

Notice:

All the inputs, outputs, and timing are the same as in Homework 4.

You only need to add the "Increase Const" functionality based on Homework 4.

Description	CMD	index	value
Increase_Const	5(101)	index	0(unused)

The functionality of "increase const" is to increase the value of heap[index] by 3.

//heap[index] <= heap[index] + 3;

3. Scoring of Question 3(b) [20%]

The scoring is fully based on the functional simulation results in Modelsim. There is no necessary to synthesis the Verilog codes. All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

(You won't receive partial credit for partially correct answers.)

```
VSIM 88> run -all
#  ****************************
#  **                        **
#  **  Congratulations !!    **
#  **                        **       |__||
#  **  Simulation PASS !!    **      / 0.0  |
#  **                        **    /_____   |
#  **  Your score =100       **   /^ ^ ^ \  |
#  **                        **  |^ ^ ^ ^ |w|
#  ****************************   \m___m__|_|
# ** Note: $finish    : C:/Users/diclab/Desktop/DIC2024/final_0531/Q3b/increase_const_1/testfixture.v(123)
#    Time: 1637500 ps  Iteration: 0  Instance: /test
```

Question 3(c): SORT CIRCUIT

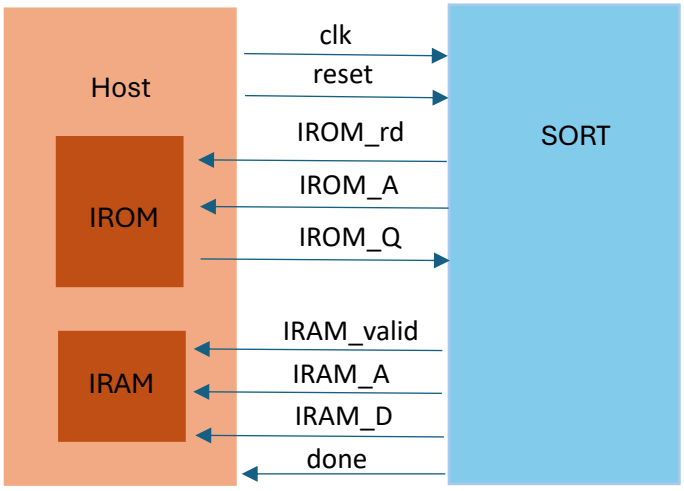
This assignment aims for you to complete the circuit design of the SORT CIRCUIT. This circuit is capable of reading data and sort data. Finally, the sorted data will be written to IRAM by ascending order. (Smallest at address 0, biggest at 15). Note: You should sort all (16) data in IROM.

1. File Description

File Name	Description
sort.v	The module of sort, which is the top module in this design.
testfixture.v	The testbench file.
pat.dat	Input Raw Data: The data format is hexadecimal.
golden.dat	Golden data for sort verification.

2. Specification:

1.1 Block Overview



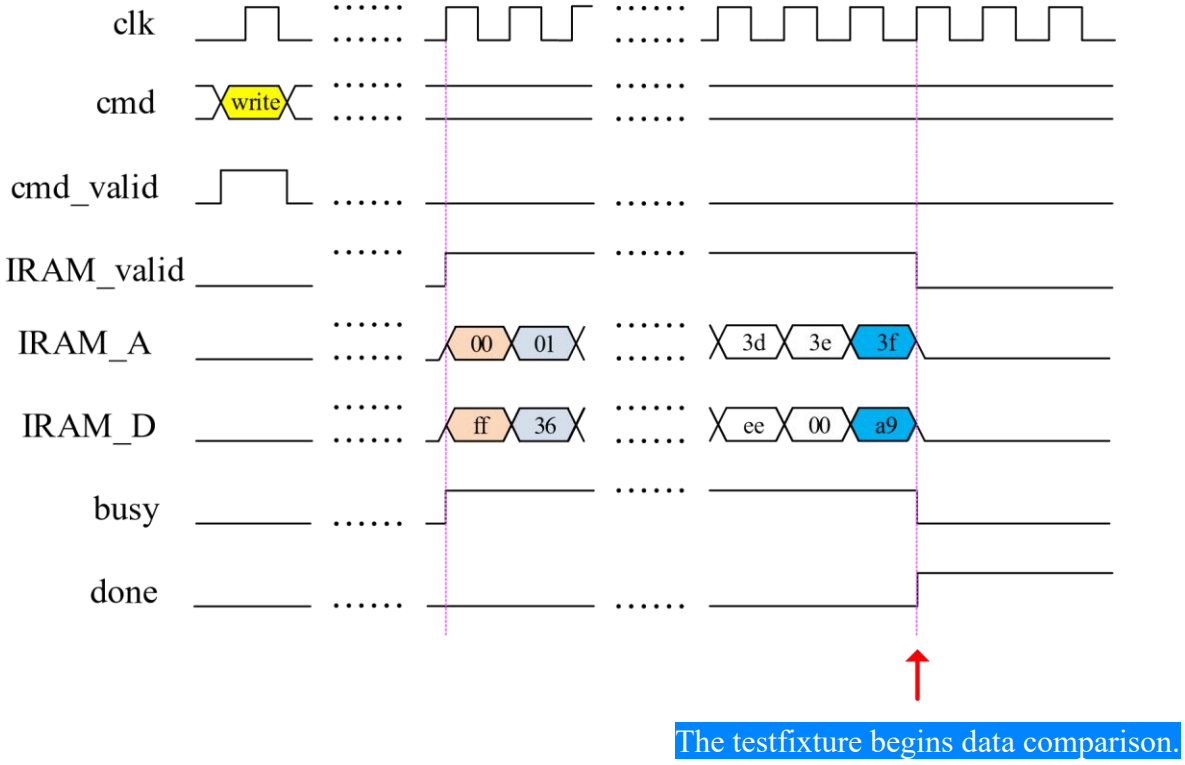
1.2 I/O Interface

Signal	I/O	Bit Width	Description
clk	input	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
reset	input	1	Active-high asynchronous reset signal.
IROM_rd	output	1	IROM read enable signal, active-high.
IROM_A	output	4	IROM Address Data. The SORT end use this output data to inform the Host end of the address to be accessed in the IROM memory.
IROM_Q	input	8	Output data which should be sorted.

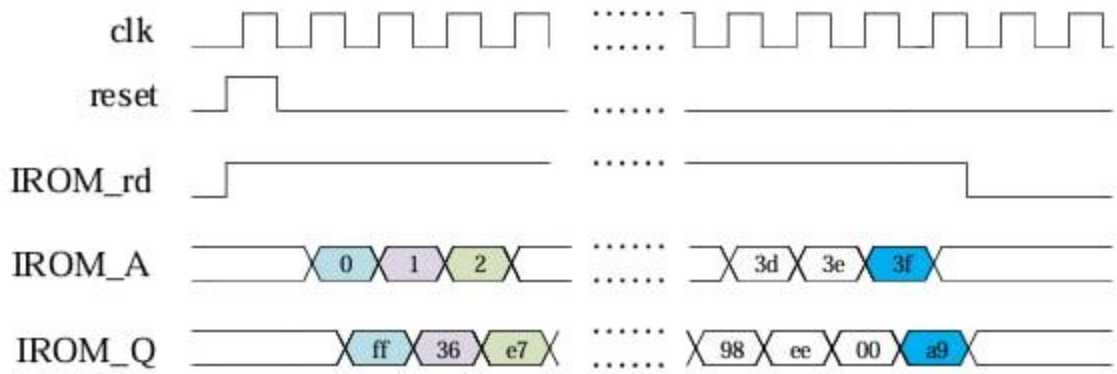
IRAM_valid	output	1	IRAM Memory Data Enable Signal: When it is high, it indicates that the data and address transmitted from the SORT end are valid.
IRAM_A	output	4	IRAM Address Data: The SORT end needs to use this output signals to instruct the Host end's RAM memory to write data to the specified address.
IRAM_D	output	8	IRAM Data: The SORT end utilizes this output signals to write data to the IRAM memory module on the Host end.
done	output	1	When the controller completes writing to RAM, setting "done" to high indicates completion.

1.3 Timing Specification Diagram

- The timing specification diagram for the write command is as shown in the following figure.
  - When executing the write command, the controller will write the processed data into the IRAM.
  - When IRAM_valid is high, it indicates writing to IRAM. At this point, address signals can be input to write image data into the IRAM.
  - After writing is completed, the "done" signal should be set to high to indicate completion. At this point, the testfixture will compare the written data in IRAM with the golden pattern.



If the IRAM_valid signal is high, data from IRAM_D will be written into address IRAM_A at the negative edge of clock.



The timing of IROM output is as shown in Figure.

Notice:

You can simply done this request by repeatedly call Extraxt_max in HW4(or Q3a) and put the extract value into RAM by ascending order (for example : Do Extraxt_max and put the extract value in RAM[15]. Do Extraxt_max again and put the extract value in RAM[14]…) until the heap is empty.

3. Scoring of Question 3(c) [10%]

The scoring is fully based on the functional simulation results in Modelsim. There is no necessary to synthesis the Verilog codes. All of the results should be generated correctly, and you will get the following message in ModelSim simulation.

(You won't receive partial credit for partially correct answers.)

```
VSIM 91> run -all
# All data have been generated successfully!
#
# --------------------PASS--------------------
#
# ** Note: $finish    : C:/Users/diclab/Desktop/DIC2024/final_0531/Q3c/testfixture.v(77)
#    Time: 1185 ns  Iteration: 0  Instance: /test
```

Submission:

1. Submitted files

You should classify your files into five directories and compress them to .zip format. The naming rule is Final_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 Final 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
Q3a	
*.v	All of your Verilog RTL code for Question 3(a)
Q3b	
*.v	All of your Verilog RTL code for Question 3(b)
Q3c	
*.v	All of your Verilog RTL code for Question 3(c)

2. Notes

Deadline: 2024/6/3 12:00

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