NYCU-EE IC LAB - Autumn 2024

Lab06 Exercise

Design: Matrix Determinant Calculator

Data Preparation

1. Extract test data from TA's directory:
   % tar xvf ~iclabTA01/Lab06.tar
2. The extracted LAB directory contains:
   a. Exercise_SoftIP/
   b. Exercise/

Design Introduction

- ***Hamming code***

Hamming code is a type of error-correcting code used to detect and correct single-bit errors in data transmission or storage. It was developed by Richard Hamming in 1950 and is widely used in computer memory systems, data transmission, and other digital communication systems where data integrity is crucial.

<u>*Example*</u>

Enccoding:

Consider an 8 bits example with data = 8'b10101111, in Hamming code, the positions that are powers of 2 (such as positions 1, 2, 4, 8, etc.) should be left empty. The original data values are then filled into the remaining positions in the sequence.

1	2	3	4	5	6	7	8	9	10	11	12
		1		0	1	0		1	1	1	1

Draw a calculation diagram where the Hamming code bit positions are arranged from highest to lowest. Then, convert the bit positions that contain "1" into binary and write them out. Then perform an XOR operation on the red values above. In fact, you only need to count how many 1s there are to determine the result. If there is an even number of 1s, the result is 0; if there is an odd number of 1s, the result is 1.

	8	4	2	1
3	0	0	1	1
6	0	1	1	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
	0	0	0	1

Now we can know our Hamming code should be 4’b0001, then we put it back to our table.

1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	1	0	0	1	1	1	1

Here we get the encode data is 12’b101001001111.

Decoding:

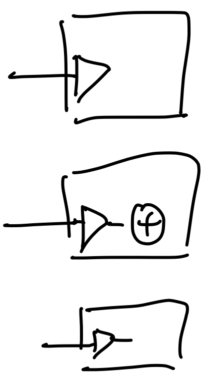
1 2 3 4 5 6 ..... IP_BIT+4

Assume we get an encoded data = 10’b1101110101, first we draw a table

1	2	3	4	5	6	7	8	9	10
1	1	0	1	1	1	0	1	0	1

IP_BIT+4-1 IP_BIT+4-2 0

Convert the positions where the bit is 1 into binary, then perform an XOR operation on these binary values.



1	0	0	0	1
2	0	0	1	0
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
8	1	0	0	0
10	1	0	1	0
	0	1	1	0

[3:0] A [IP_BIT+4 : 1]

A[1] = (I_C[8] == 1

A[8]

The resulting value is 0110, which is 6 in decimal, indicating that there is an error in the 6th bit. If there were no errors, the result would be 0000.

Change the 6th bit of the original value from 1 to 0 (or vice versa).

1101110101 $\rightarrow$ 1101100101

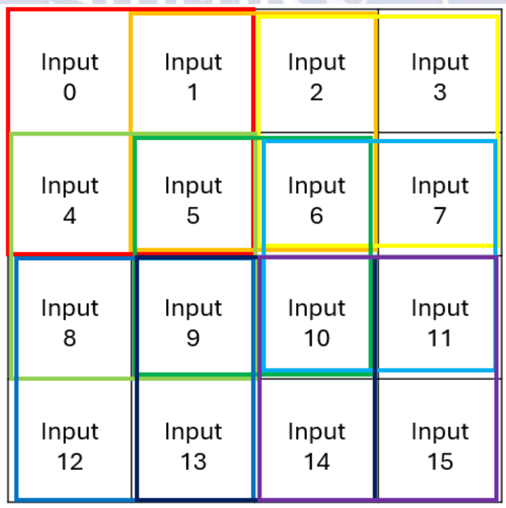
Then we can get the correct value.

■ ***Matrix Determinant***

In this lab, we are going to do 3 different metrix size determinant, which are 2*2, 3*3, and 4*4. You will be giving 16 input data and an input code to decide which kind of size we want to calculate. And the input order is like below fig.

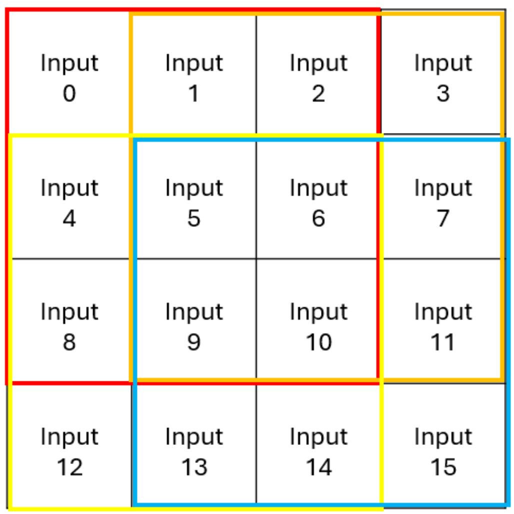
Input 0	Input 1	Input 2	Input 3
Input 4	Input 5	Input 6	Input 7
Input 8	Input 9	Input 10	Input 11
Input 12	Input 13	Input 14	Input 15

When doing 2*2 matrix determinant, we will have to calculate 9 different output like below fig.



(out 0 = 0,1,4,5, out 1 = 1,2,5,6...)

When doing 3*3 matrix determinant, we will have to calculate 4 different output like below fig.



(out 0 = 0,1,2,4,5,6,8,9,10 ...)

And last, we have 4*4 matrix determinant, there's only one output like below fig.

Input 0	Input 1	Input 2	Input 3
Input 4	Input 5	Input 6	Input 7
Input 8	Input 9	Input 10	Input 11
Input 12	Input 13	Input 14	Input 15

- ***Calculation***

You will recieve a "in_valid" signal for 16 cycle, also a sequence "in_data" (16 input in series), each is 15 bits, encoded with 15-11 hamming code. And also, an additional input "in_mode", each is 9 bit, encoded with 9-5 hamming code. First, you need to decode those data to get the real data and insrtuction, then according to the instruction to do the right calculation. Notice that there might have error bit you need to correct, but only one bit will be wrong. Finally, "out_valid" will be raised for one cycle if you finish all calculation, and "out_data" will output your calcualtion result. Be sure "out_valid" should be high for only one cycle and cannot be overlaped with in_valid.

	Input bit width	After decode bit width
in_valid	1 bit	-
in_data	15 bits	11 bits (signed)
in_mode	9 bits	5 bits

Insruction (decoded)	Matrix size	Output
5'b00100	2*2	9 output
5'b00110	3*3	4 output
5'b10110	4*4	1 output

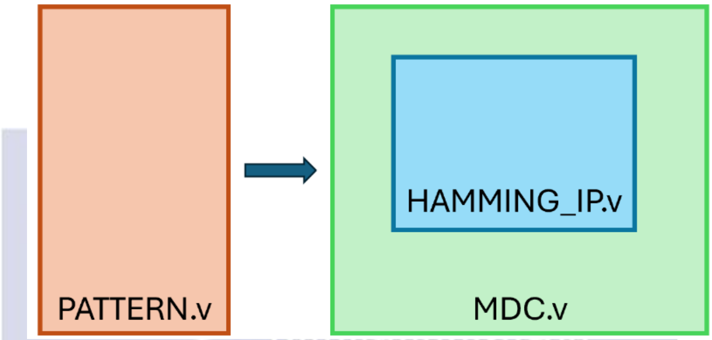
<table>
<tr><td>Size</td><td colspan="45">out_data (207 bits)</td></tr>
<tr><td></td><td colspan="45">MSB … LSB</td></tr>
<tr><td>2*2</td><td colspan="5">out 0
23 bits</td><td colspan="5">out 1
23bits</td><td colspan="5">out 2
23 bits</td><td colspan="5">out 3
23 bits</td><td colspan="5">out 4
23 bits</td><td colspan="5">out 5
23 bits</td><td colspan="5">out 6
23 bits</td><td colspan="5">out 7
23 bits</td><td colspan="5">out 8
23 bits</td></tr>
<tr><td>3*3</td><td colspan="9">None
3 bit</td><td colspan="9">out 0
51 bits</td><td colspan="9">out 1
51 bits</td><td colspan="9">out 2
51 bits</td><td colspan="9">out 3
51 bits</td></tr>
<tr><td>4*4</td><td colspan="45">out 0
207 bits</td></tr>
</table>

Design Description

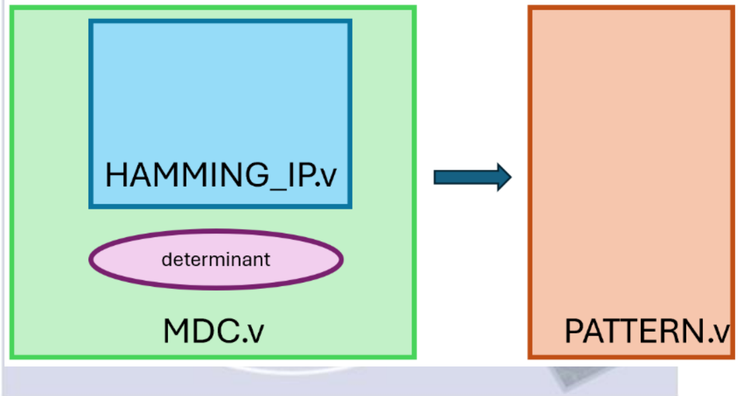
Design	Definition
Soft IP	Given input data. The soft IP needs to decode data.
Top Design	Given input data, and mode. The top design needs to calculate the different size of matrix determinant and output the result.

- ***Soft IP***
- ***Top Design***

Step-1: Given input data, and mode. Use your own designed soft IP to decode data and mode.



Step-2: Use the decoded data and mode to calculate determinant.



Inputs and Outputs (Top Design)

- The following are the definitions of input signals:

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_data	15	The data needs to be decoded and used to calculate determinant.
in_mode	9	If decoded in_mode is 00100, output 9 2*2 determinant result. If decoded in_mode is 00110, output 4 3*3 determinant result. If decoded in_mode is 10110, output 1 4*4 determinant result.

■ The following are the definitions of output signals:

Output signal	Bit width	Definition
out_valid	1	High when output signals are valid.
out_code	207	Output the corresponded determinant result.

Inputs and Outputs (Soft IP)

■ The following are the definitions of input signals:

Input signal	Bit width	Definition
IN_code	IP_BIT + 4	The data that need to be decoded.

9 ~ 15

■ In soft ip demo, IN_code is a data you need to decode. Notice that there might have error bit to be corrected. And if there is an error bit, it will only be one bit at a time.

■ The following are the definitions of two parameters:

Parameter	Bit width	Definition
IP_BIT	-	The length of original data (un-encoded). IP_BIT will be range in 5-11.

Example: #(4)

1. 10 means that there is a ten bits data before encode.
2. **There won't be illegal case #(0)**.

■ The following are the definitions of output signals:

Output signal	Bit width	Definition
OUT_code	IP_BIT	Output of the decoded data

Specifications (Top Design)

Top module

1. Top module name: **MDC** (design file name: **MDC.v**).
2. You can adjust your clock period by yourself, but the maximum period is 20ns. The precision of clock period is 0.1, for example, 40.5 is allowed, but 40.55 is not allowed.
3. The execution latency is limited to 1000 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**.
4. The total cell area should not be larger than 1,000,000 um^2.
5. The look-up table method is forbidden, and you need to use your own-designed soft IP (HAMMING_IP) in the top module. (TA will check your design)

Reset

6. 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.

7. 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.

in valid

8. **in_valid** will come after reset.
9. All input signals are synchronized at **negative edge** of the clock.
10. When **in_valid** is low, input is tied to unknown state.

out valid

11. **out_valid** should not be raised when **in_valid** is high.
12. The **out_valid** is limited to be high only when the output value is valid.
13. All output signals should be synchronized at clock positive edge.
14. The TA's pattern will capture your output for checking at **clock negative edge**.
15. The **out_code** should be correct when **out_valid** is high.
16. The next input pattern will come in 2~4 **negative edge of clock** after your **out_valid** falls.

Synthesis

17. In this lab, you should write your own **syn.tcl file**.
18. Use **top** wire load mode and **compile_ultra**.
19. Use analyze + elaborate to read your design.
20. The input delay is set to 0.5*(clock period).
21. The output delay is set to 0.5*(clock period), and the output loading is set to 0.05.
22. The input delay of clk and rst_n should be zero.
23. The synthesis time should be less than 1 hours.
24. The synthesis result (syn.log) of data type **cannot** include any latches and error.
25. After synthesis, you can check **MDC.area** and **MDC.timing**. The area report is valid when the slack in the end of timing report should be **non-negative** and the result should be **MET**.

Gate level simulation

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

Supplement

27. In this lab, you are NOT allowed to use Designware IP.
28. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *pass*, *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.
29. Don't write chinese comments or other language comments in the file you turned in.

30. 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.

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.

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.

Specifications (Soft IP)

Top module

1. Top module name: **HAMMING_IP** (design file name: **HAMMING _IP.v**)
   Input signals: **IN_code**
   Output signals: **OUT_code**
   One parameters : **IP_BIT**
2. The clock period is 20ns. Finish calculating within **one** clock cycle.
3. The look-up table method is forbidden. (TA will check your design)

Supplement

4. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *pass*, *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.
5. Don't write chinese or other language comments in the file you turned in.
6. 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.

   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.

   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.

Note

1. **Grading policy:**
   - Function Validity: 50% (RTL and Gate-level simulation correctness)
     - Top design
   - Performance: 30% $Area^2 * Total\ Latency * Cycle\ time$
     - Top design
   - Soft IP function correctness: 20% (No second demo)
     - IP_WIDTH = 11-bits: 3%
     - IP_WIDTH = 10-bits: 3%
     - IP_WIDTH = 9-bits: 3%
     - IP_WIDTH = 8-bits: 3%
     - IP_WIDTH = 7-bits: 3%
     - IP_WIDTH = 6-bits: 3%
     - IP_WIDTH = 5-bits: 2%
   - The performance is determined by area and latency of your design. The less cost your design has, the higher grades you get.
   - The grade of 2nd demo would be 30% off.
   - Latency is the execution latency plus 1. If out_valid rises immediately after in_valid falls, the latency is 1. Total Latency is the sum of the latency for each pattern.

2. **Please submit your files under 09_SUBMIT. (09_SUBMIT** is under **Exercise/.)**

   1st demo : before 12:00 p.m. on 10/21(Mon.)

   2nd demo : before 12:00 p.m. on 10/23(Wed.)

   You should check the following files under **09_SUBMIT/Lab06_iclabXXX/**

   - Top : MDC_iclabxxx.v
     Soft IP : HAMMING_IP_iclabxxx.v
     Cycle Time : CYCLE_iclabxxx.txt
     Syn.tcl : syn_iclabxxx.tcl
     Filelist : filelist_iclabxxx.f
   - xxx is your account number, i.e. MDC_iclab999.v、HAMMING_IP_iclab999.v
   - If you miss any files on the list, you will fail this lab.
   - If the uploaded file violating the naming rule, you will get 5 deduct points.

   Then use the command like the figure below to check the files are uploaded or not.

```
[Exercise/09_SUBMIT]% ./02_check 1st_demo
```

3. **Template folders and reference commands:**

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

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

   (Check latch by searching the keyword “Latch” in syn.log)

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

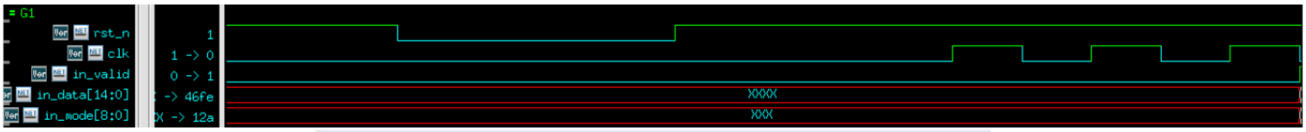
(Check the design's area in /Report/ MDC.area)

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

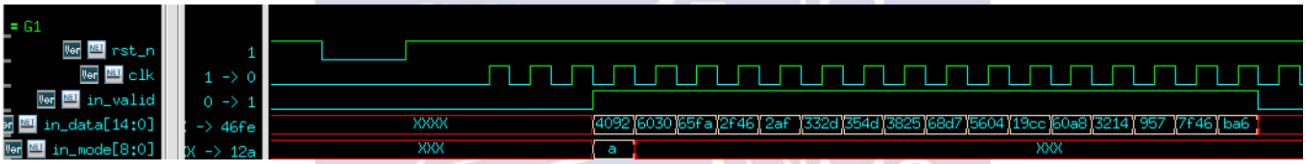
- You can key in **./09_clean** to clear all log files and dump files in each folder.
- You should make sure the **three clock period values identical** in 00_TESTBED/PATTERN.v and 02_SYN/syn.tcl

Sample Waveform

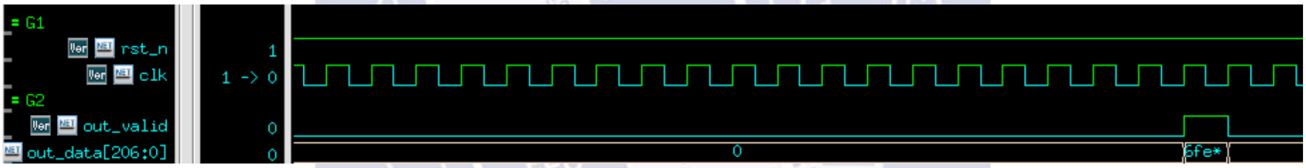
4. Asynchronous reset and active-low and reset all output.



5. 16 cycle for input the information of data and 1 cycle for input the information of mode in each round/pattern.



6. Output the result of deferent mode in each round/pattern.



- The **out_data** should be correct when **out_valid** is high, that is, **both signals will be checked at every clock negative edge**.