Lab 1: Gate-Level Verilog

Group 3 : 109060013 張芯瑜 109062328 吳邦寧

Design Explanation

1. DMUX

Essentially, dmux is a selector. AND gate can be seen as a signal input and a selection input, that is, selection input could cut off the signal input. Taking advantage of the characteristic of AND gate, we shall design our DMUX.

In our design, we build a small dmux(sdmux) module first. Sdmux is use to choose the correct input for each bit of the output. Second, we check the select signal to decide which input is our goal. Last, we throw the select signal and inputs into sdmux to get our final answer. The details are down below:

(1) Design Specification:

For sdmux module

Input in (each bit of the input)

Input a, b, c, d (selection inputs)

Output A, B, C, D (outputs for every bits)

For dmux module

Input [3:0] in

Input [1:0] sel

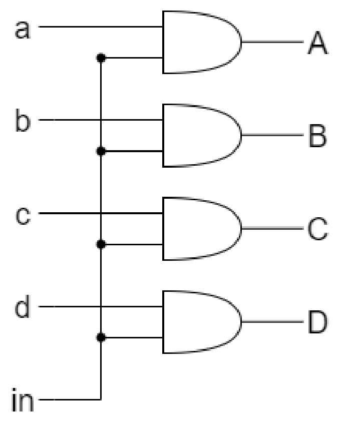
Output [3:0] a, b, c, d

(2) Truth table:

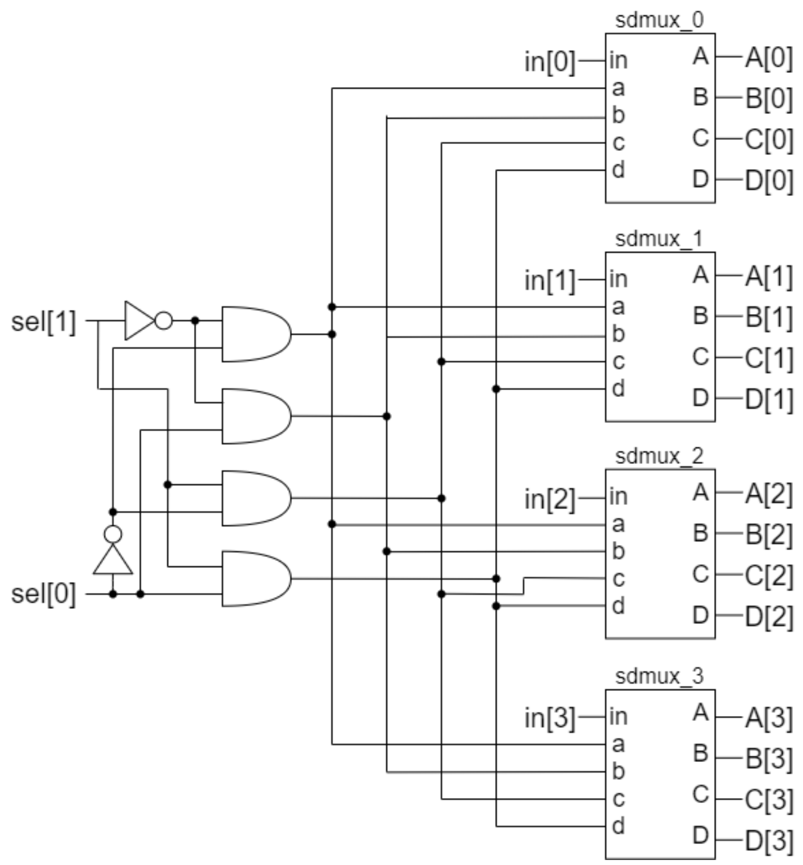
[3:0] in	[1:0] sel	[3:0] a	[3:0] b	[3:0] c	[3:0] d
x	00	[3:0] in	0000	0000	0000
x	01	0000	[3:0] in	0000	0000
x	10	0000	0000	[3:0] in	0000
x	11	0000	0000	0000	[3:0] in

(3) Gate Level Circuits:

- SDMUX



- DMUX



2. Crossbar 2x2 4bit

First, we build both mux and demux module of 1 bit. Mux is simply a selector, so taking advantage of AND gate's selection characteristics would make the design easier. Demux is the inverse function of Mux, thus AND gate is also handy when it comes to designing.Second, we implement the circuit by the schematic design. The details are down below:

(1) Design Specification:

For dmux_1_2 module

Input sel

Input [3:0] in

Output [3:0] out1, out2

For mux_2_1 module

Input sel

Input [3:0] in1, in2

Output [3:0] out

For 4-bit crossbar2x2 module

Input control

Input [3:0] in1, in2

Output [3:0] out1, out2

(2) Truth Table:

For dmux_1_2 module

in	sel	out1	out2
x	0	in	0
x	1	0	in

For mux_2_1 module

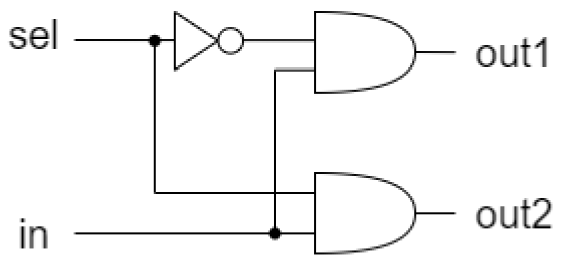
in1	in2	sel	out
x	x	0	in1
x	x	1	in2

For 4-bit crossbar module

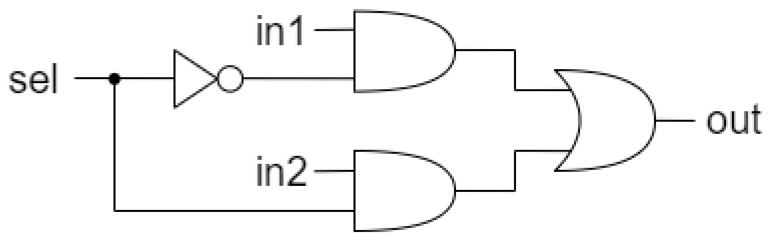
[3:0] in1	[3:0] in2	control	[3:0] out1	[3:0] out2
x	x	0	[3:0] in2	[3:0] in1
x	x	1	[3:0] in1	[3:0] in2

(3) Gate Level Circuits:

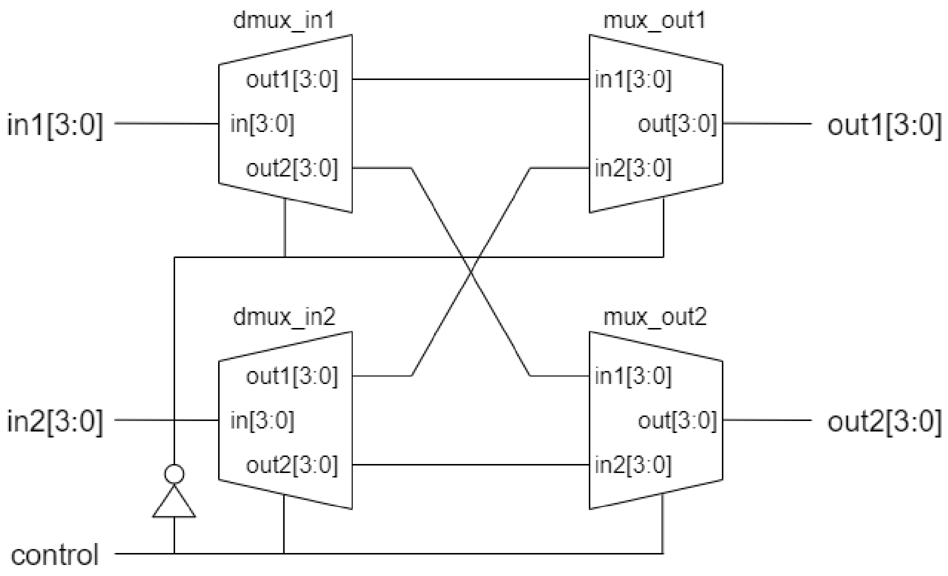
- DMUX_1_2



- MUX_2_1



- Crossbar 2x2



3. Crossbar 4x4 4bit

We use the Crossbar2x2 module from the previous question, and build the Crossbar4x4. The details are down below:

(1) Design Specification:

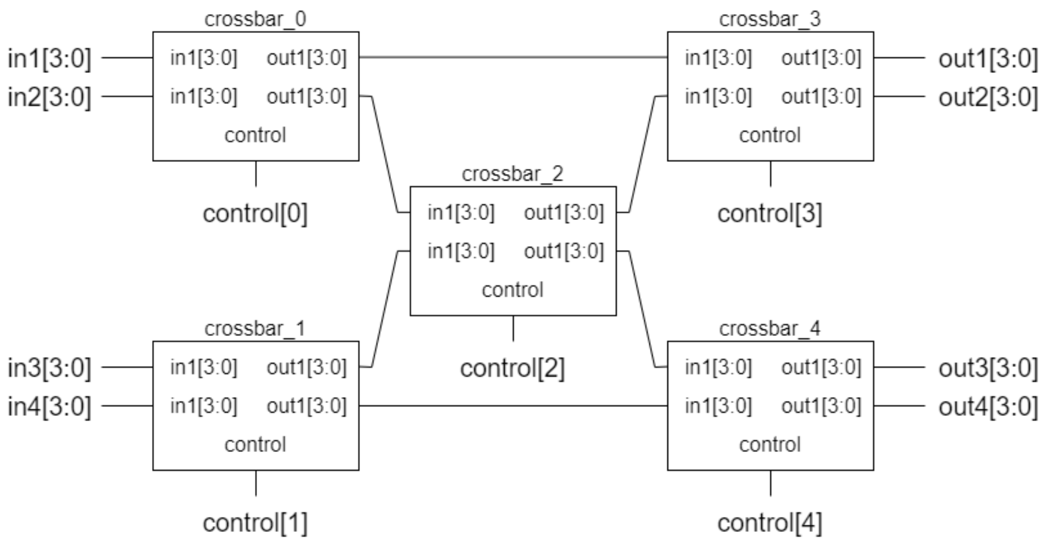
For 4-bit crossbar4x4 module

Input [4:0] control

Input [3:0] in1, in2, in3, in4

Output [3:0] out1, out2, out3, out4

(2) Gate Level Circuits:



4. Toggle Flip Flop

By using the D_Flip_Flop from the basic question, we build a T_Flip_Flop. The details are down below:

(1) Design Specification:

For 1-bit T_Flip_Flop

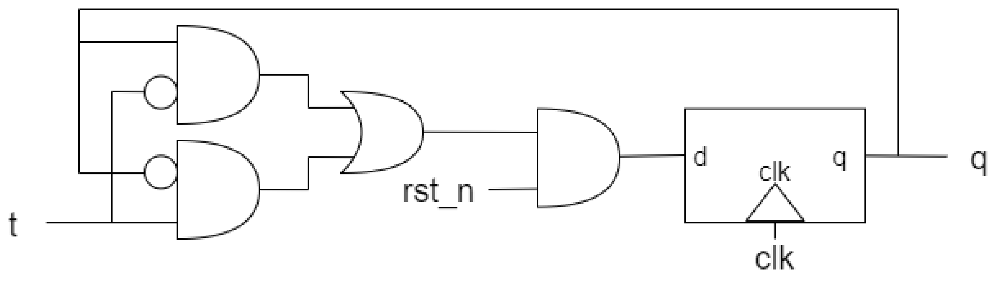
Input rst_n, t, clk

Output q

(2) Truth Table:

rst_n	t	q_next
0	0	0
0	1	0
1	0	q
1	1	~q

(3) Gate Level Circuits:



5. FPGA (4-bit simple crossbar)

In this part, we use the same module as question2, which called crossbar2x2. Since there are 2 LEDs for each output, we add two output signal in our design. The details are down below:

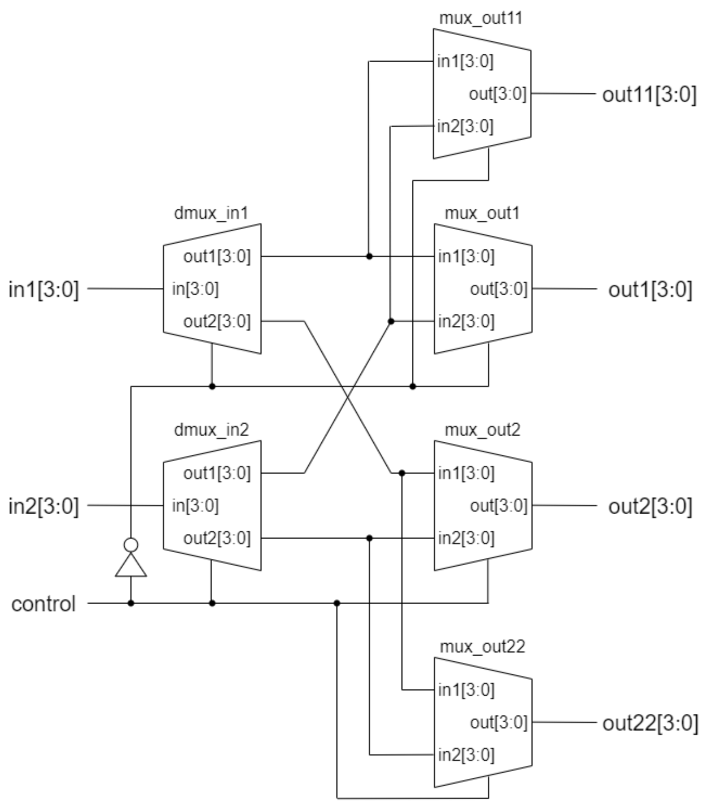
(1) Design Specification:

Input control (for switch)

Input [3:0] in1, in2 (for switch)

Output [3:0] out1, out11, out2, out22 (for LED)

(2) Gate Level Circuits:



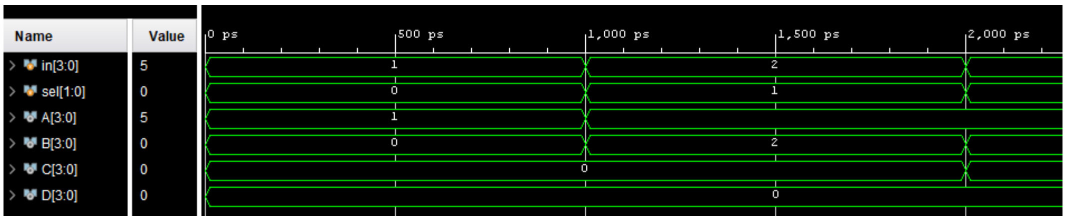
(3) I/O Pin Assignment:

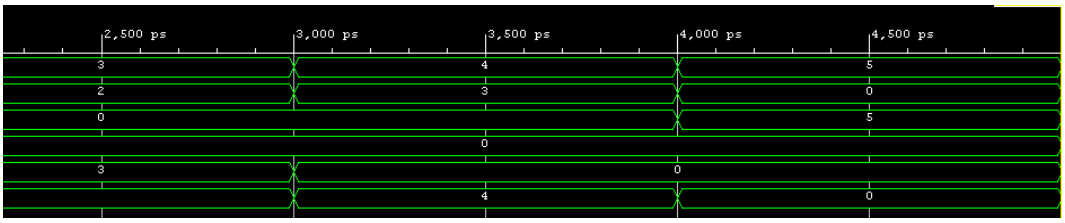
I/O	control	in1[0]	in1[1]	in1[2]	in1[3]	in2[0]
LOC	V17	V16	W16	W17	W15	V15
I/O	in2[1]	in2[2]	in2[3]	out1[0]	out1[1]	out1[2]
LOC	W14	W13	V2	U16	U19	W18
I/O	out1[3]	out11[0]	out11[1]	out11[2]	out11[3]	out2[0]
LOC	U14	E19	V19	U15	V14	V13
I/O	out2[1]	out2[2]	out2[3]	out22[0]	out22[1]	out22[2]
LOC	W3	P3	P1	V3	U3	N3
I/O	out22[3]					
LOC	L1					

Design Verification

1. DMUX

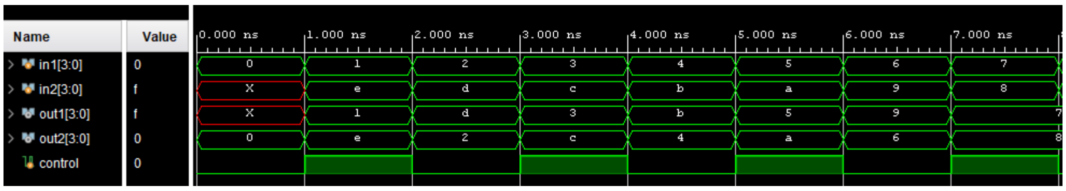
Enumerate all possible inputs, and verify the outputs manually. We initial ‘in’ to 1 and ‘sel’ to 0 at first. Then add 1 to both of them each step. We repeat it for 4 times to check does the all kinds of ‘sel’ goes well. The pictures below are the representation with verification.

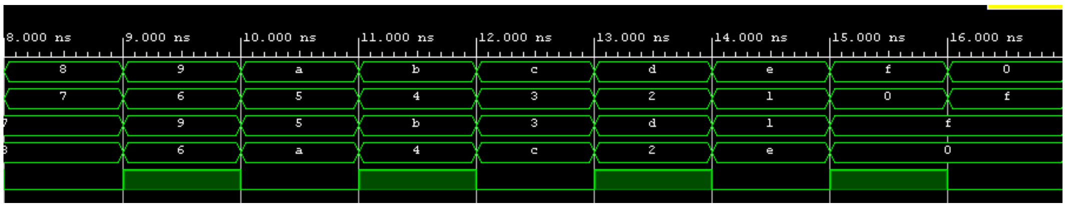




2. Crossbar 2x2 4bit

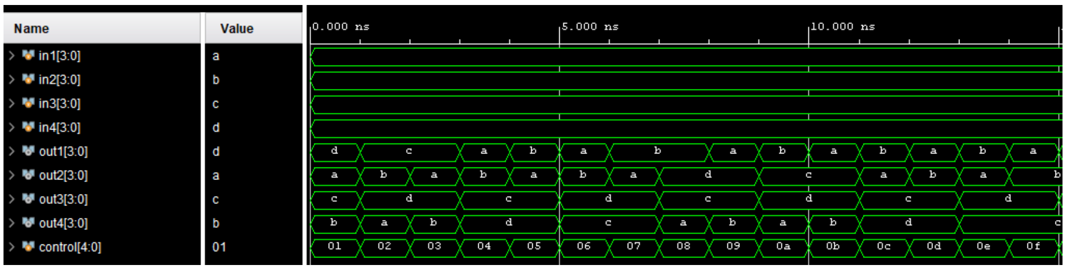
Initially, let ‘in1’ be zero. Also, always let ‘in2’ be the complement of in2. While enumerating ‘in1’, ‘control’ would be flipped every step. Last, we verify the output waves manually. The pictures below are the representation with verification.

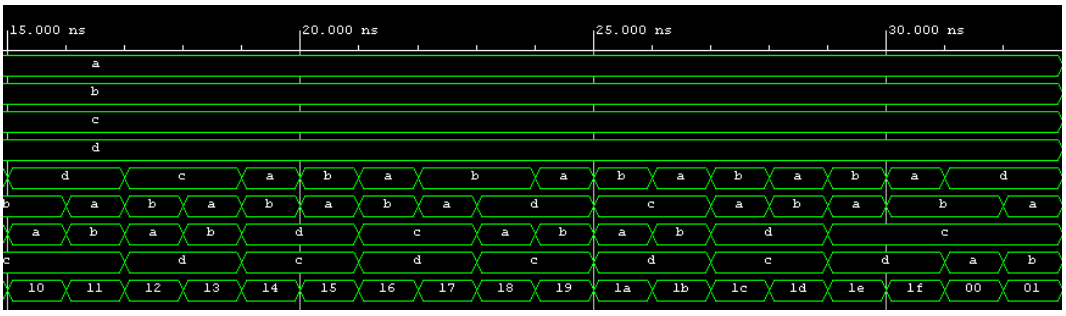




3. Crossbar 4x4 4bit

Initiallly, let ‘in1’ be a, ‘in2’ be b, ‘in3’ be c, and ‘in4’ be d. Then add 1 to ‘control’ every step while ‘control’ is initialed as 0. Last, we verify the output waves manually. The pictures below are the representation with verification.



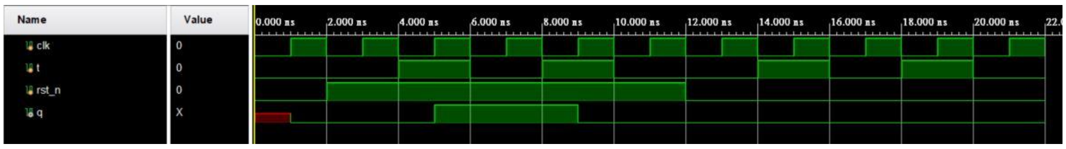


In above result, we can also found out that groups 'cdba', 'cdab', 'dcab', 'dcba' are not routable as the inputs are 'abcd' for in1 to in4. So we can tell the result below never appears.

	[3:0] out1	[3:0] out2	[3:0] out3	[3:0] out4
1	[3:0] in3	[3:0] in4	[3:0] in1	[3:0] in2
2	[3:0] in3	[3:0] in4	[3:0] in2	[3:0] in1
3	[3:0] in4	[3:0] in3	[3:0] in1	[3:0] in2
4	[3:0] in4	[3:0] in3	[3:0] in2	[3:0] in1

4. Toggle Flip Flop

We switch 't' every clock and check the results. Also, we initial 'rst_n' as 1 and changed it to 0 after 5 clocks to check if our module goes well. Last, we verify the output waves manually. The picture below is the representation with verification.



Contribution

1. Lawrence Wu

Design & Verification of DMUX and Crossbar 2x2.

Collaboration environment setup.

Report writing of DMUX and Crossbar 2x2.

Found an open sourced project 'Yosys' for synthesis schematic graphs.

2. Ariel Chang

   Design & Verification of Crossbar 4x4 and Toggle Flip Flop.

   Report writing of Crossbar 4x4 and Toggle Flip Flop.

   Typesetting the whole report after both of us are done with our parts.

   Learned how to use ‘draw.io’ to make graphs and made lots of beautiful design diagrams.

What have we learned?

Initial the variables at the beginning are important!!!!

Verification is usually a harder than designing chips.

Mapping the PIN is crucial when it comes to demonstrate.

Don’t use English.

We have learned how to output Verilog to FPGA.

We have learned how to let FPGA lights up.

0-stands for light down, 1-stands for lights up.