**Hardware Design and Lab: Lab1**

**111060013 EECS 26' 劉祐廷**

1. **Crossbar\_2x2\_4bit**
2. **Design Specification**

* **Input**

**[3:0] in1** is a four-bit data.

**[3:0] in2** is a four-bit data.

**control** is a signal to control the crossbar.

* **Output**

**[3:0] out1** is a four-bit data.

**[3:0] out2** is a four-bit data.

* **Wire**

**not\_control** is the invert of control.

**[3:0] in1\_out1** is a four-bit wire connecting **DMux1.out0** and **Mux1.in0**.

**[3:0] in1\_out2** is a four-bit wire connecting **DMux1.out1** and **Mux2.in0**.

**[3:0] in2\_out1** is a four-bit wire connecting **DMux2.out0** and **Mux1.in1**.

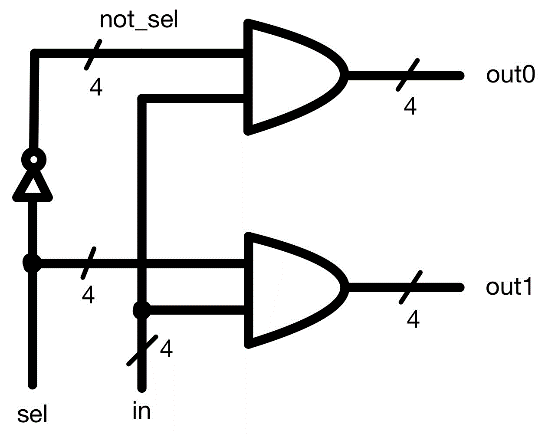
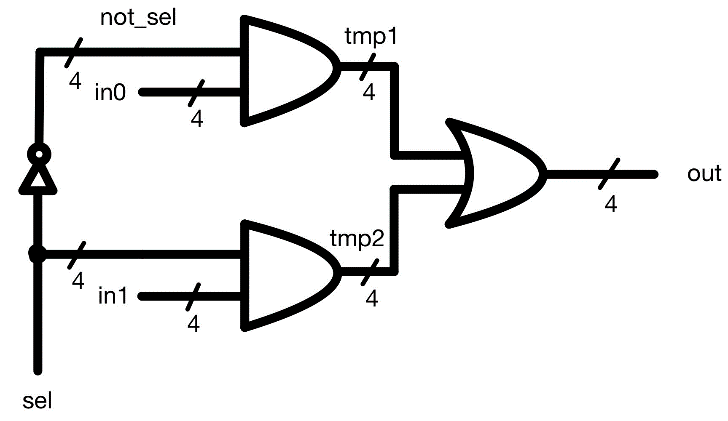
**[3:0] in2\_out2** is a four-bit wire connecting **DMux2.out1** and **Mux2.in1**.

* **Module**

**Mux\_2to1\_4bits:** Mux1, Mux2

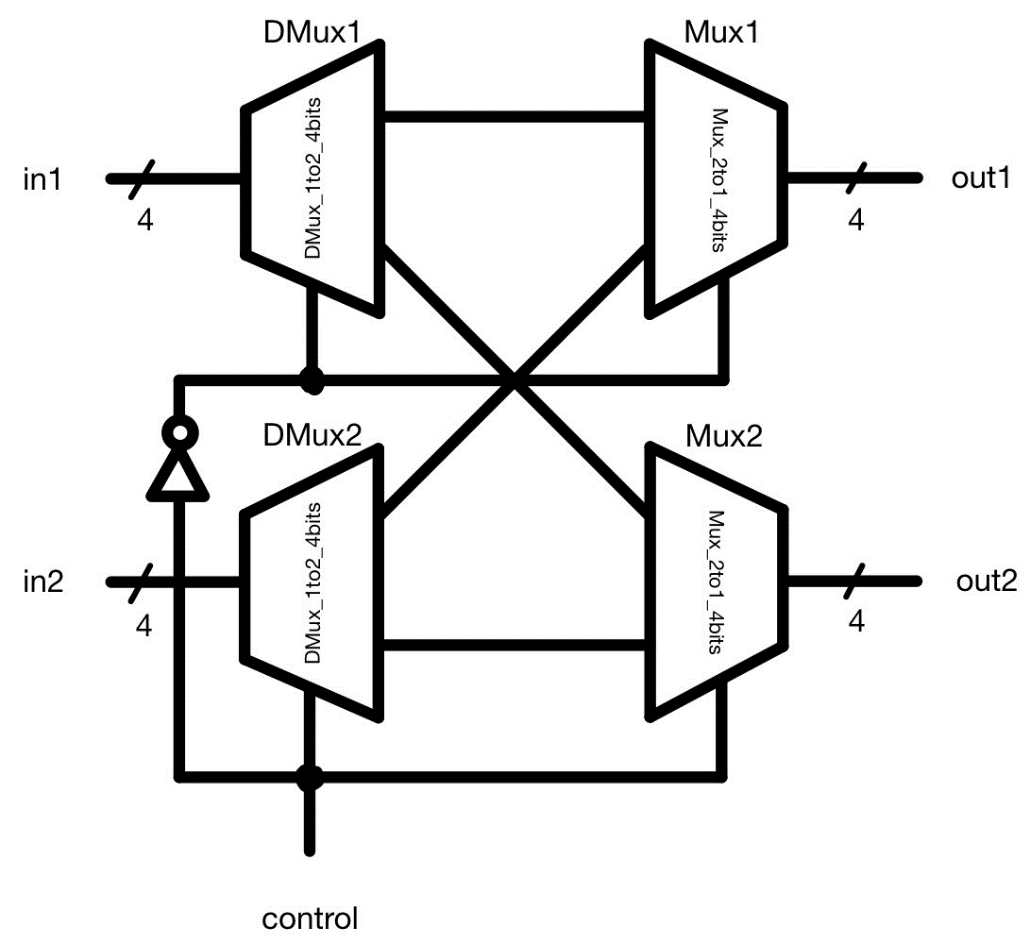
**DMux\_1to2\_4bits:** DMux1, DMux2

1. **Block Diagram**



**Mux\_2to1\_4bits Circuit**

**DMux\_1to2\_4bits Circuit**



**Crossbar\_2x2\_4bits Circuit**

1. **Explanation**

**Crossbar\_2x2\_4bit** module consists of two types of modules, **DMux\_1to2\_4bits** module and **Mux\_2to1\_4bits** module. For **DMux\_1to2\_4bits** module, I used a NOT gate to get the invert of **sel**, which is called **not\_sel**. And then I connected them respectively to two different AND gates. We know that in Boolean algebra ***1 & X = X***. From this theorem, I used AND gates to control which output port the data would go out. For **Mux\_2to1\_4bits** module, I used AND gates to control which input the Mux would choose to output and used OR gates to combine the output data because in Boolean algebra, ***1 & X = X*** and ***0 | X = X***. After that, I defined **DMux1**, **DMux2**, **Mux1** and **Mux2** and connected them as the block diagram above to implement that if **control** is 1, **in1** will be passed to **out1** and **in2** will be passed to **out2**. Otherwise, **in1** will be passed to **out2** and **in2** will be passed to **out1**.

To test the design, I've written a test bench to change input data and switch control so that I can check if the output meets the expectation or not.

1. **Crossbar\_4x4\_4bit**
2. **Design Specification**

* **Input**

**[3:0] in1** is a four-bit data.

**[3:0] in2** is a four-bit data.

**[3:0] in3** is a four-bit data.

**[3:0] in4** is a four-bit data.

**[4:0] control** is a five-bit signal to control the crossbar.

* **Output**

**[3:0] out1** is a four-bit data.

**[3:0] out2** is a four-bit data.

**[3:0] out3** is a four-bit data.

**[3:0] out4** is a four-bit data.

* **Wire**

**[3:0] CB0\_CB1** is a four-bit wire connecting **CB0.out0** and **CB1.in0**.

**[3:0] CB0\_CB2** is a four-bit wire connecting **CB0.out1** and **CB2.in0**.

**[3:0] CB2\_CB1** is a four-bit wire connecting **CB2.out0** and **CB1.in1**.

**[3:0] CB3\_CB2** is a four-bit wire connecting **CB3.out0** and **CB2.in1**.

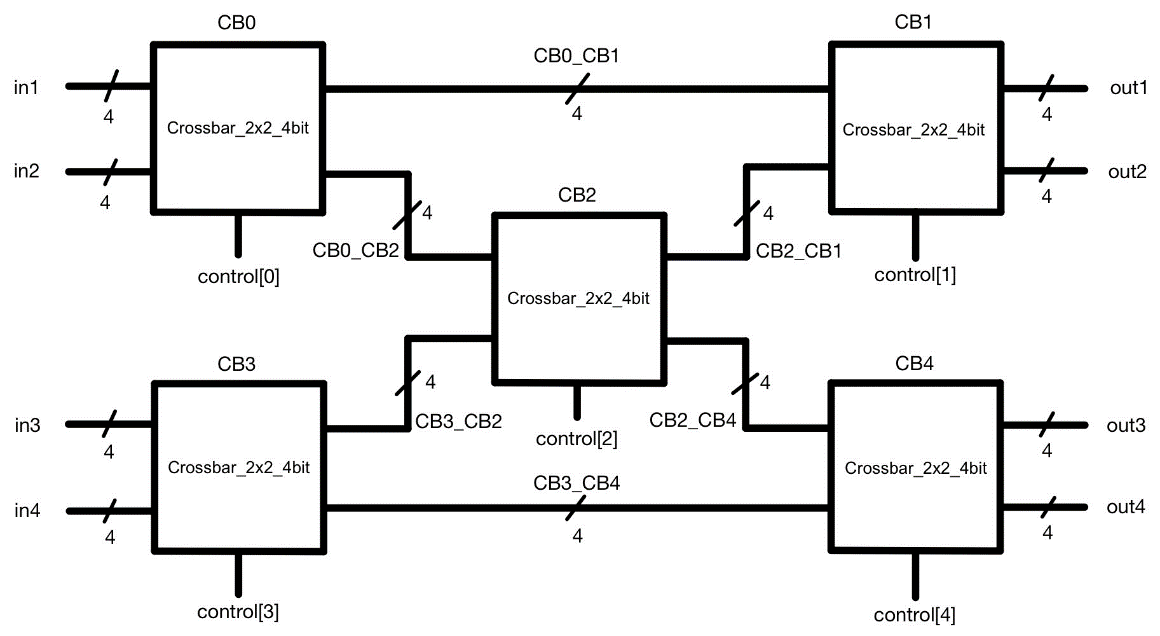
**[3:0] CB2\_CB4** is a four-bit wire connecting **CB2.out1** and **CB4.in0**.

**[3:0] CB3\_CB4** is a four-bit wire connecting **CB4.out1** and **CB4.in1**.

* **Module**

**Crossbar\_2x2\_4bits:** CB0, CB1, CB2, CB3, CB4

1. **Block Diagram**



**Crossbar\_4x4\_4bit Circuit**

1. **Explanation**

**Crossbar\_4x4\_4bit** module consists of five **Crossbar\_2x2\_4bit** modules. These five modules are be connected as the block diagram above and all of them are implemented by same method mentioned in **1. Crossbar\_2x2\_4bit – C. Explanation**.

To test the design, I fixed the values of the four inputs and change the **control** from **5'b00000** to **5'b11111** to check whether the results are correct or not.

1. **Crossbar\_2x2\_4bits\_fpga**
2. **Design Specification**

* **Input**

**[3:0] in1** is a four-bit data.

**[3:0] in2** is a four-bit data.

**control** is a signal to control the crossbar.

* **Output**

**[3:0] out1** is a four-bit data.

**[3:0] out1\_extra** is a four-bit data.

**[3:0] out2** is a four-bit data.

**[3:0] out2\_extra** is a four-bit data.

* **Wire**

**not\_control** is the invert of control.

**[3:0] in1\_out1** is a four-bit wire connecting **DMux1.out0** and **Mux1.in0**.

**[3:0] in1\_out2** is a four-bit wire connecting **DMux1.out1** and **Mux2.in0**.

**[3:0] in2\_out1** is a four-bit wire connecting **DMux2.out0** and **Mux1.in1**.

**[3:0] in2\_out2** is a four-bit wire connecting **DMux2.out1** and **Mux2.in1**.

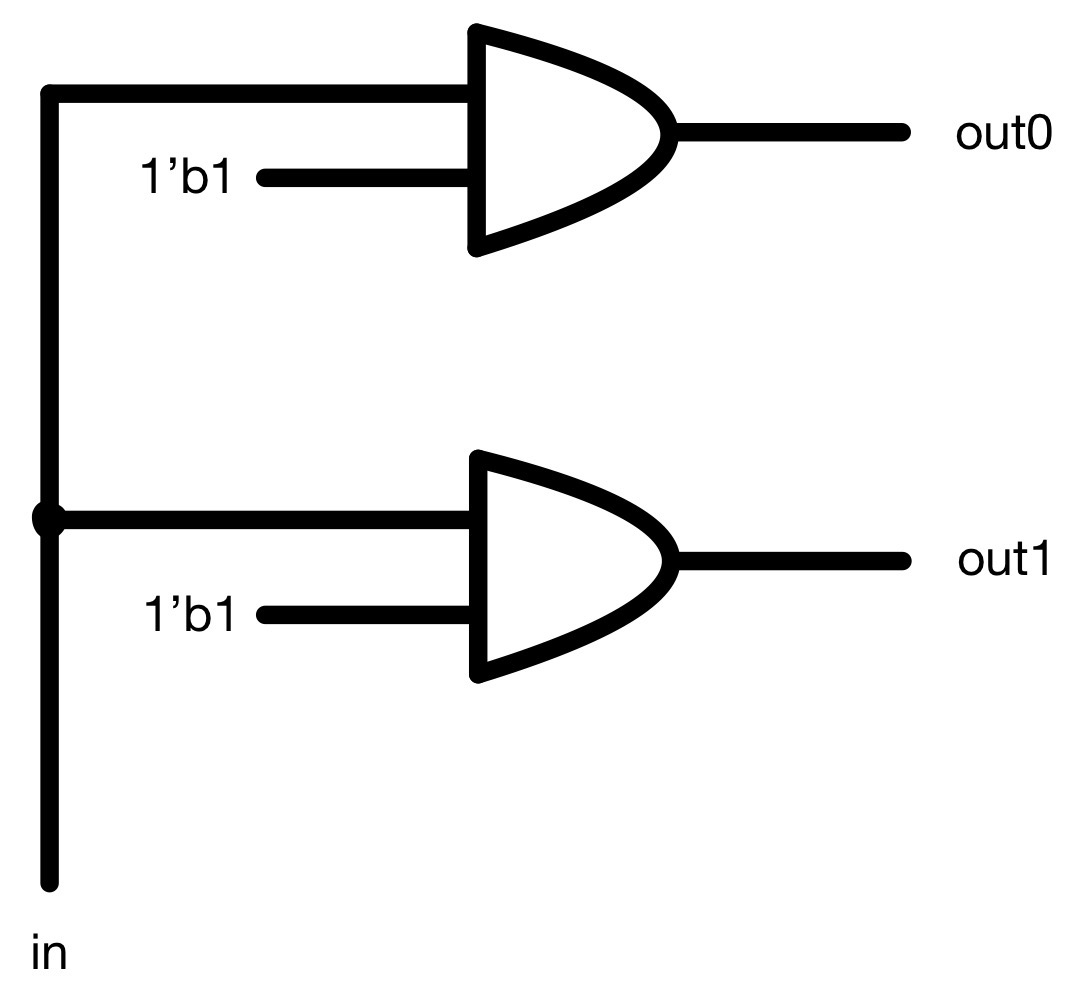
* **Module**

**Fanout\_1to2\_4bits:** FO1, FO2

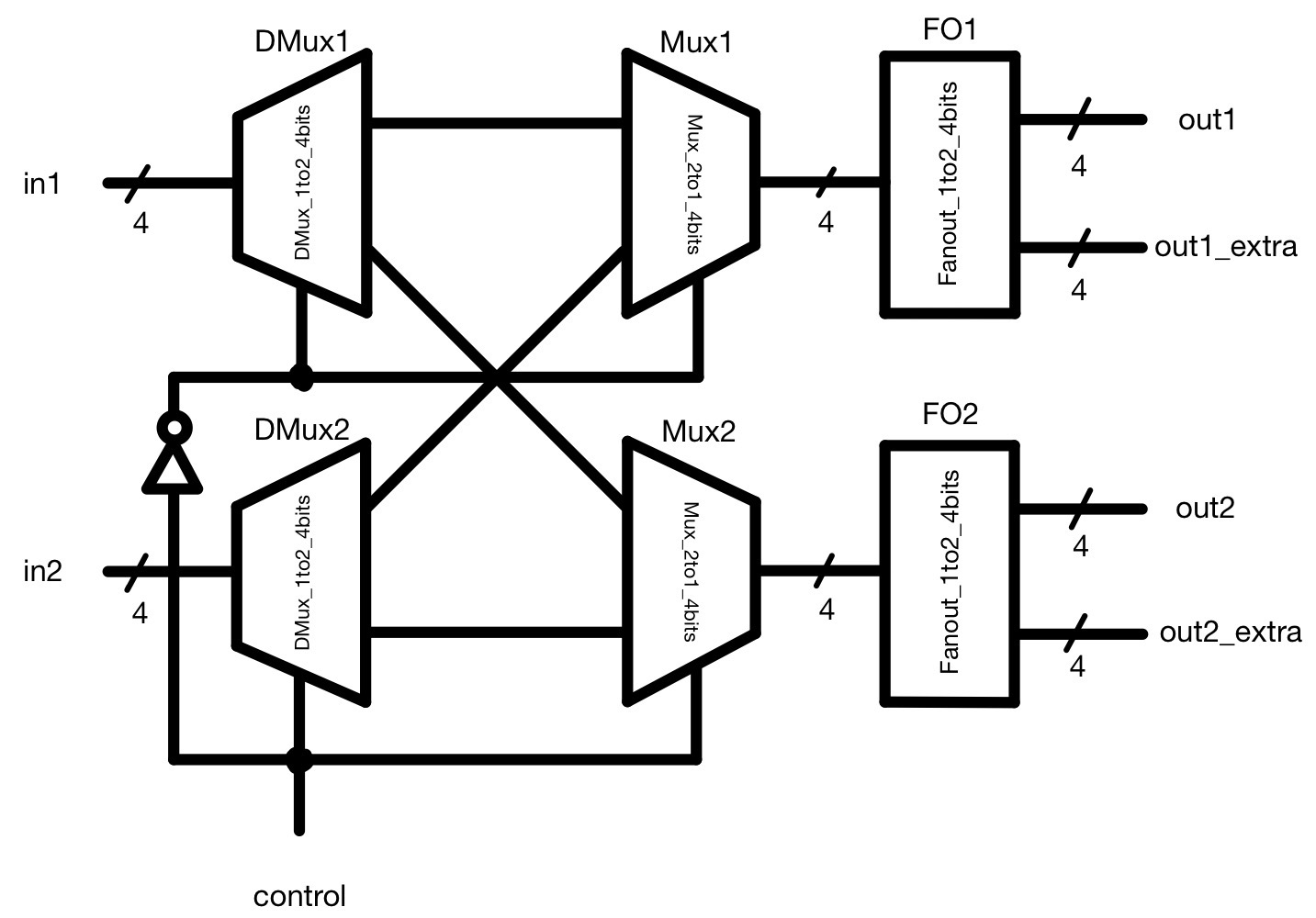
**Mux\_2to1\_4bits:** Mux1, Mux2

**DMux\_1to2\_4bits:** DMux1, DMux2

1. **Block Diagram**



**Fanout\_1to2\_4bits Circuit**



**Crossbar\_2x2\_4bit\_fpga Circuit**

1. **Explanation**

Most of this module is same as the **Crossbar\_2x2\_4bit** module in **1. Crossbar\_2x2\_4bit**. The difference is that this module connects to two **Fanout\_1to2\_4bits** modules in order to link to two LED on FPGA board for each output.

To test this design, I programed it on FPGA board and turn the switch on and off to check every situation of input and output is correct.

1. **Discussion**

In Lab 1, I've learned the difference of thinking ways between software design and hardware design. Hardware design is more like putting blocks together while software design is more like dealing with some different events. Through this lab, I've got more familiar to Vivado and FPGA board. I hope that these experiences can help me learn well in the following classes.