邏輯系統實驗 實驗四

組別：7

成員：章子嚴、張軒、魏晉成

學號：E24065018、E24066129、E24066226

實驗內容：

一、基礎題(一)：四對一多工器

四對一多工器中，我們使用了 3 個二對一多工器來組合，首先在 sel[0]先選擇要 input A/C、或是 input B/D 哪一組，接著再用 sel[1]去選擇上述選擇結果的其中之一，便能達到以 sel[1:0]的 00/01/10/11 組合去選擇到 A/B/C/D 了。

而以下我放上我們的二對一、四對一多工器以及 test bench 的 verilog code，還有 test bench 輸出以及波形的截圖。

```
module mux_2to1(in0, in1, sel, out);

    input    in0, in1, sel;
    output out;

    wire Not_Out0, And_Out0, And_Out1;//Not_Out0 is the ~sel, And_Out0 is
          ~sel*in0, And_Out1 is sel*in1

    not NOT0(Not_Out0, sel);
    and AND0(And_Out0, in0, Not_Out0);
    and AND1(And_Out1, in1, sel);
    or    OR0(out, And_Out0, And_Out1);

endmodule
```

(二對一多工器)

```
module mux_4to1(in, sel, out);

    input [3:0]in;
    input [1:0]sel;
    output out;

    wire M2T1_out[0:1];

    mux_2to1 M2T1_0(in[0], in[1], sel[0], M2T1_out[0]);
    mux_2to1 M2T1_1(in[2], in[3], sel[0], M2T1_out[1]);
    mux_2to1 M2T1_2(M2T1_out[0], M2T1_out[1], sel[1], out);


endmodule
```

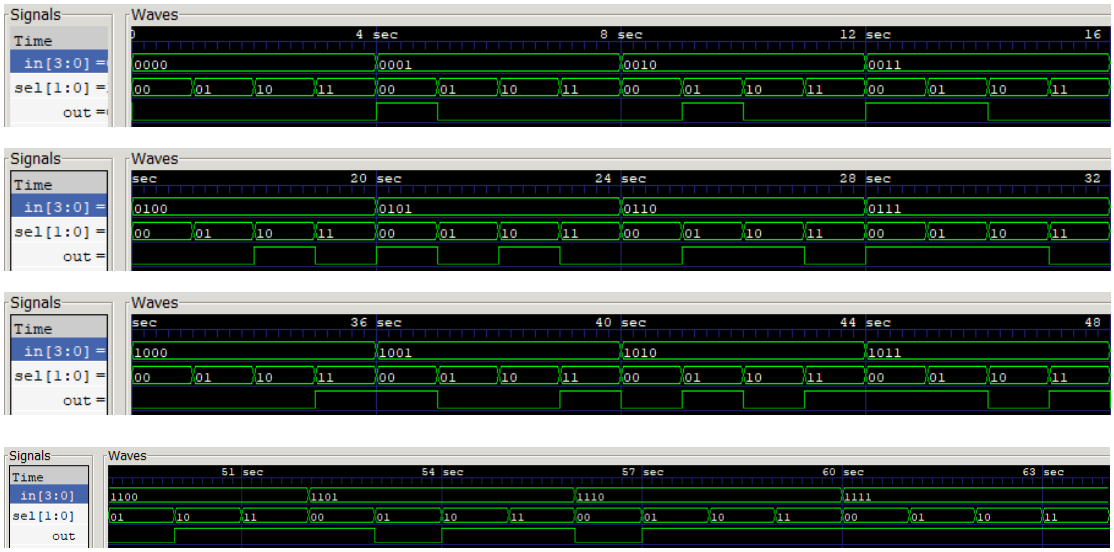
(四對一多工器)

```
module test;
	reg [3:0]in;
	reg [1:0] sel;

	wire out;

	mux_4to1 M4T1(in, sel, out);

	initial begin
		for(in=4'd0; in<15; in=in+1)begin
			for(sel=2'd0; sel<3; sel=sel+1)begin
				#1;
			end
			#1 sel=sel+1;
		end

		for(sel=2'd0; sel<3; sel=sel+1)begin
			#1;
		end
		#1 sel=sel+1;
	end

	initial begin
		$monitor("in[3:0]= %b, sel=%b, out=%b", in, sel, out);
	end

	initial begin
		$dumpfile("testM4T1.vcd");
		$dumpvars;
	end

endmodule
```

(test bench)

in[3:0]= 0000, sel=00, out=0	in[3:0]= 1000, sel=00, out=0
in[3:0]= 0000, sel=01, out=0	in[3:0]= 1000, sel=01, out=0
in[3:0]= 0000, sel=10, out=0	in[3:0]= 1000, sel=10, out=0
in[3:0]= 0000, sel=11, out=0	in[3:0]= 1000, sel=11, out=1
in[3:0]= 0001, sel=00, out=1	in[3:0]= 1001, sel=00, out=1
in[3:0]= 0001, sel=01, out=0	in[3:0]= 1001, sel=01, out=0
in[3:0]= 0001, sel=10, out=0	in[3:0]= 1001, sel=10, out=0
in[3:0]= 0001, sel=11, out=0	in[3:0]= 1001, sel=11, out=1
in[3:0]= 0010, sel=00, out=0	in[3:0]= 1010, sel=00, out=0
in[3:0]= 0010, sel=01, out=1	in[3:0]= 1010, sel=01, out=1
in[3:0]= 0010, sel=10, out=0	in[3:0]= 1010, sel=10, out=0
in[3:0]= 0010, sel=11, out=0	in[3:0]= 1010, sel=11, out=1
in[3:0]= 0011, sel=00, out=1	in[3:0]= 1011, sel=00, out=1
in[3:0]= 0011, sel=01, out=1	in[3:0]= 1011, sel=01, out=1
in[3:0]= 0011, sel=10, out=0	in[3:0]= 1011, sel=10, out=0
in[3:0]= 0011, sel=11, out=0	in[3:0]= 1011, sel=11, out=1
in[3:0]= 0100, sel=00, out=0	in[3:0]= 1100, sel=00, out=0
in[3:0]= 0100, sel=01, out=0	in[3:0]= 1100, sel=01, out=0
in[3:0]= 0100, sel=10, out=1	in[3:0]= 1100, sel=10, out=1
in[3:0]= 0100, sel=11, out=0	in[3:0]= 1100, sel=11, out=1
in[3:0]= 0101, sel=00, out=1	in[3:0]= 1101, sel=00, out=1
in[3:0]= 0101, sel=01, out=0	in[3:0]= 1101, sel=01, out=0
in[3:0]= 0101, sel=10, out=1	in[3:0]= 1101, sel=10, out=1
in[3:0]= 0101, sel=11, out=0	in[3:0]= 1101, sel=11, out=1
in[3:0]= 0110, sel=00, out=0	in[3:0]= 1110, sel=00, out=0
in[3:0]= 0110, sel=01, out=1	in[3:0]= 1110, sel=01, out=1
in[3:0]= 0110, sel=10, out=1	in[3:0]= 1110, sel=10, out=1
in[3:0]= 0110, sel=11, out=0	in[3:0]= 1110, sel=11, out=1
in[3:0]= 0111, sel=00, out=1	in[3:0]= 1111, sel=00, out=1
in[3:0]= 0111, sel=01, out=1	in[3:0]= 1111, sel=01, out=1
in[3:0]= 0111, sel=10, out=1	in[3:0]= 1111, sel=10, out=1
in[3:0]= 0111, sel=11, out=0	in[3:0]= 1111, sel=11, out=1

(test bench 的結果)



(波形圖)

舉波形圖的第 36 秒為例，當 input=1001 的時候，sel=00，選擇到了 A(也就是 input[0])，因此 out 呈現 1 的狀態。

二、基礎題(二)：全加器

在這個實驗中，我們利用兩個半加器以及一個產生 Carry Out 的 OR 閘來組合出一個全加器，因此稍後會放上半加器、全加器以及 test bench 的 verilog code，以及全加器的 test bench 測試輸出與波形圖。

```
module semiAdder(in0, in1, sum, carry);
	input in0, in1;
	output sum, carry;

	xor XOR0(sum, in0, in1);
	and AND0(carry, in0, in1);

endmodule
```

(半加器)

```
module fullAdder(in0, in1, in2, sum, carry);

	input in0, in1, in2;
	output sum, carry;

	wire carry0, carry1, sum0;

	semiAdder SA0(in0, in1, sum0, carry0);
	semiAdder SA1(in2, sum0, sum, carry1);
	or OR0(carry, carry0, carry1);

endmodule
```

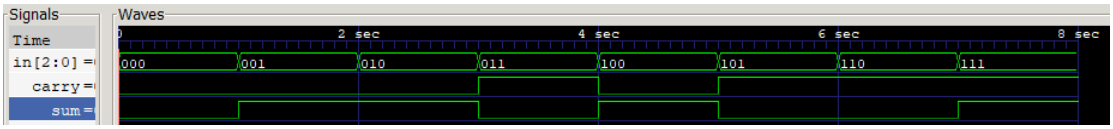
(全加器)

```
module testFullAdder;

	reg [2:0] in;

	wire sum;
	wire carry;

	fullAdder FA0 (.in0(in[0]),.in1(in[1]),.in2(in[2]),.sum(sum),.carry(carry));

	initial begin
		for(in=3'd0; in<7; in=in+1)begin
			#1;
		end
		#1;
	end

	initial begin
		$monitor("in0=%b, in1=%b, in2=%b, sum=%b, carry=%b", in[0], in[1],
in[2], sum, carry);
	end

	initial begin
		$dumpfile("testFullAdder.vcd");
		$dumpvars;
	end

endmodule
```

(test bench)

```
in0=0, in1=0, in2=0, sum=0, carry=0
in0=1, in1=0, in2=0, sum=1, carry=0
in0=0, in1=1, in2=0, sum=1, carry=0
in0=1, in1=1, in2=0, sum=0, carry=1
in0=0, in1=0, in2=1, sum=1, carry=0
in0=1, in1=0, in2=1, sum=0, carry=1
in0=0, in1=1, in2=1, sum=0, carry=1
in0=1, in1=1, in2=1, sum=1, carry=1
```

(test bench 輸出)



(波形輸出)

以第 6 秒的波形為例，輸入的三個 bit 分別為 110，而 $1+1+0=10_{(2)}$，所以 carry 呈現高電位，sum 呈現低電位。

三、挑戰題：4bits 漣波加法器

利用四個全加器模組，將較低權重的(右側的)加法器 carry out 與 較高權重的(左側的)加法器 carry in 相接，便能得到一個由右側傳至左側的漣波加法器。

而以下是漣波加法器與其 test bench 的 Verilog code(半加器與全加器的 verilog code 在基礎題(二)中)；至於模擬數據以及波形的部分，由於兩個 4 bit input 加上 1 bit 的 carry in 總共有 512 種組合，故僅擷取部分結果放入結報中。

```
module fourBitFA(INa, INb, CarryIn, Sum, CarryOut);

	input [3:0]INa;
	input [3:0]INb;
	input CarryIn;
	output [3:0]Sum;
	output CarryOut;

	wire FA_out[0:2];

	fullAdder FA0(CarryIn, INa[0], INb[0], Sum[0], FA_out[0]);
	fullAdder FA1(FA_out[0], INa[1], INb[1], Sum[1], FA_out[1]);
	fullAdder FA2(FA_out[1], INa[2], INb[2], Sum[2], FA_out[2]);
	fullAdder FA3(FA_out[2], INa[3], INb[3], Sum[3], CarryOut);

endmodule
```

(4bit 全加器 Verilog code)

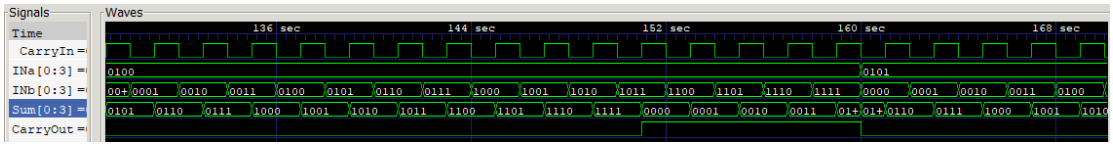
```
module testFourBitFA;

	reg [0:3] INa;
	reg [0:3] INb;
	reg CarryIn;

	wire [0:3] Sum;
	wire CarryOut;

	fourBitFA fourBitFA0
(.INa(INa),.INb(INb),.CarryIn(CarryIn),.Sum(Sum),.CarryOut(CarryOut));

	initial begin
		for(INa=0; INa<15; INa=INa+1)begin
			for(INb=0; INb<15; INb=INb+1)begin
				CarryIn=0;
				#1 CarryIn=1;
				#1;
			end
			CarryIn=0;
			#1 CarryIn=1;
			#1;
		end
		for(INb=0; INb<15; INb=INb+1)begin
				CarryIn=0;
				#1 CarryIn=1;
				#1;
		end
		CarryIn=0;
		#1 CarryIn=1;
		#1;
	end
```

```
        initial begin
            $monitor(" CarryIn=%b, INa=%b, INb=%b, Sum=%b, CarryOut=%b",
CarryIn, INa, INb, Sum, CarryOut);
        end

        initial begin
            $dumpfile("testFourBitFA.vcd");
            $dumpvars;
        end

endmodule
```

(test bench 的 verilog code)

```
CarryIn=1, INa=0001, INb=0011, Sum=0101, CarryOut=0
CarryIn=1, INa=0010, INb=0001, Sum=0100, CarryOut=0
CarryIn=0, INa=0010, INb=0010, Sum=0100, CarryOut=0
CarryIn=1, INa=0010, INb=0010, Sum=0101, CarryOut=0
CarryIn=0, INa=0010, INb=0011, Sum=0101, CarryOut=0
CarryIn=1, INa=0011, INb=1100, Sum=0000, CarryOut=1
CarryIn=0, INa=0011, INb=1101, Sum=0000, CarryOut=1
CarryIn=1, INa=0011, INb=1101, Sum=0001, CarryOut=1
CarryIn=0, INa=0011, INb=1110, Sum=0001, CarryOut=1
```

(test bench 產生的部分 Output)



(隨機找一段波形擷取)

拿” CarryIn=0, INa=0010, INb=0010, Sum=0100, CarryOut=0” 這行結果而言，INa+INb+ CarryIn =2+2+0=4，{CarryOut, Sum}=4=00100；再取” CarryIn=1, INa=0011, INb=1101, Sum=0001, CarryOut=1”這段結果來說，INa+INb+CarryIn=3+7+1=11，{CarryOut, Sum}=11=10001。

四、心得：

張軒：Verilog 果然很炫炮，啊可是這是不是代表我以後每學一個新東西就要再學一個新程式語言啊(No~~~~~~~)...好在 Verilog 的資料處理層次寫起來好像還真的跟 C 有點像，倒是邏輯閘層次就有點麻煩，要像實際在接線路一樣一個一個弄好，有點納悶既然有資料處理層次為什麼還要邏輯閘層次呀?這樣如果電路很大的話邏輯閘層次不是要寫到死掉嗎?

章子嚴：这次的实验，我大概学会了怎么写 verilog，之前上课都听得不是很懂，实作让我比较容易懂，也学会了如何用 Xilinx ISE。

魏晉成：因為在丙班正課中，老師的第一份作業就是要求我們要以 verilog 的邏輯閘層次以及資料處理層次分別寫一個全加器，以及它的 test bench，因此多少對於 verilog 這種東西有點理解；不過單純認為用 Xilinx 的 ISE 有些麻煩，畢竟電腦小，容量有點不足。

而下次的實驗居然能寫到 ALU，讓我小小期待了一下，因為這是以前在計算機概論上聽過的東西，而如今我們卻真的要把它實做出來，令我還蠻興奮的。