邏輯系統實驗 實驗五

組別：7

成員：章子嚴、張軒、魏晉成

學號：E24065018、E24066129、E24066226

實驗內容：

1. 基礎題(一)：4bit 加法器：

在此實驗中，我們利用資料處理層次語法寫出4 bit加法器模組，在模擬器上模擬後，再燒錄至FPGA進行驗證。

以下是verilog code、testbench、模擬後的結果以及驗證結果。

module fourBitFA(inA, inB, carryIn, sum, carryOut);

input [3:0]inA, inB;

input carryIn;

output [3:0]sum;

output carryOut;

assign {carryOut, sum}=inA+inB+carryIn;

endmodule

(4 bit 加法器的verilog code)

(4 bit 加法器的test bench)

`timescale 1ns / 1ps

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

endmodule

(4 bit加法器模擬結果-節錄)

CarryIn=0, INa=0000, INb=0000, Sum=0000, CarryOut=0

CarryIn=1, INa=0000, INb=0000, Sum=0001, CarryOut=0

CarryIn=0, INa=0000, INb=0001, Sum=0001, CarryOut=0

CarryIn=1, INa=0000, INb=0001, Sum=0010, CarryOut=0

CarryIn=0, INa=0000, INb=0010, Sum=0010, CarryOut=0

CarryIn=1, INa=0000, INb=0010, Sum=0011, CarryOut=0

CarryIn=0, INa=0000, INb=0011, Sum=0011, CarryOut=0

CarryIn=1, INa=0000, INb=0011, Sum=0100, CarryOut=0

CarryIn=0, INa=0000, INb=0100, Sum=0100, CarryOut=0

CarryIn=1, INa=0000, INb=0100, Sum=0101, CarryOut=0

CarryIn=0, INa=0000, INb=0101, Sum=0101, CarryOut=0

CarryIn=1, INa=0000, INb=0101, Sum=0110, CarryOut=0

CarryIn=0, INa=0000, INb=0110, Sum=0110, CarryOut=0

CarryIn=1, INa=0000, INb=0110, Sum=0111, CarryOut=0

CarryIn=0, INa=0000, INb=0111, Sum=0111, CarryOut=0

CarryIn=1, INa=0000, INb=0111, Sum=1000, CarryOut=0

CarryIn=0, INa=0000, INb=1000, Sum=1000, CarryOut=0

CarryIn=1, INa=0000, INb=1000, Sum=1001, CarryOut=0

CarryIn=0, INa=0000, INb=1001, Sum=1001, CarryOut=0

CarryIn=1, INa=0000, INb=1001, Sum=1010, CarryOut=0

CarryIn=0, INa=0000, INb=1010, Sum=1010, CarryOut=0

CarryIn=1, INa=0000, INb=1010, Sum=1011, CarryOut=0

CarryIn=0, INa=0000, INb=1011, Sum=1011, CarryOut=0

CarryIn=1, INa=0000, INb=1011, Sum=1100, CarryOut=0

CarryIn=0, INa=0000, INb=1100, Sum=1100, CarryOut=0

CarryIn=1, INa=0000, INb=1100, Sum=1101, CarryOut=0

CarryIn=0, INa=0000, INb=1101, Sum=1101, CarryOut=0

CarryIn=1, INa=0000, INb=1101, Sum=1110, CarryOut=0

CarryIn=0, INa=0000, INb=1110, Sum=1110, CarryOut=0

CarryIn=1, INa=0000, INb=1110, Sum=1111, CarryOut=0

CarryIn=0, INa=0000, INb=1111, Sum=1111, CarryOut=0

CarryIn=1, INa=0000, INb=1111, Sum=0000, CarryOut=1

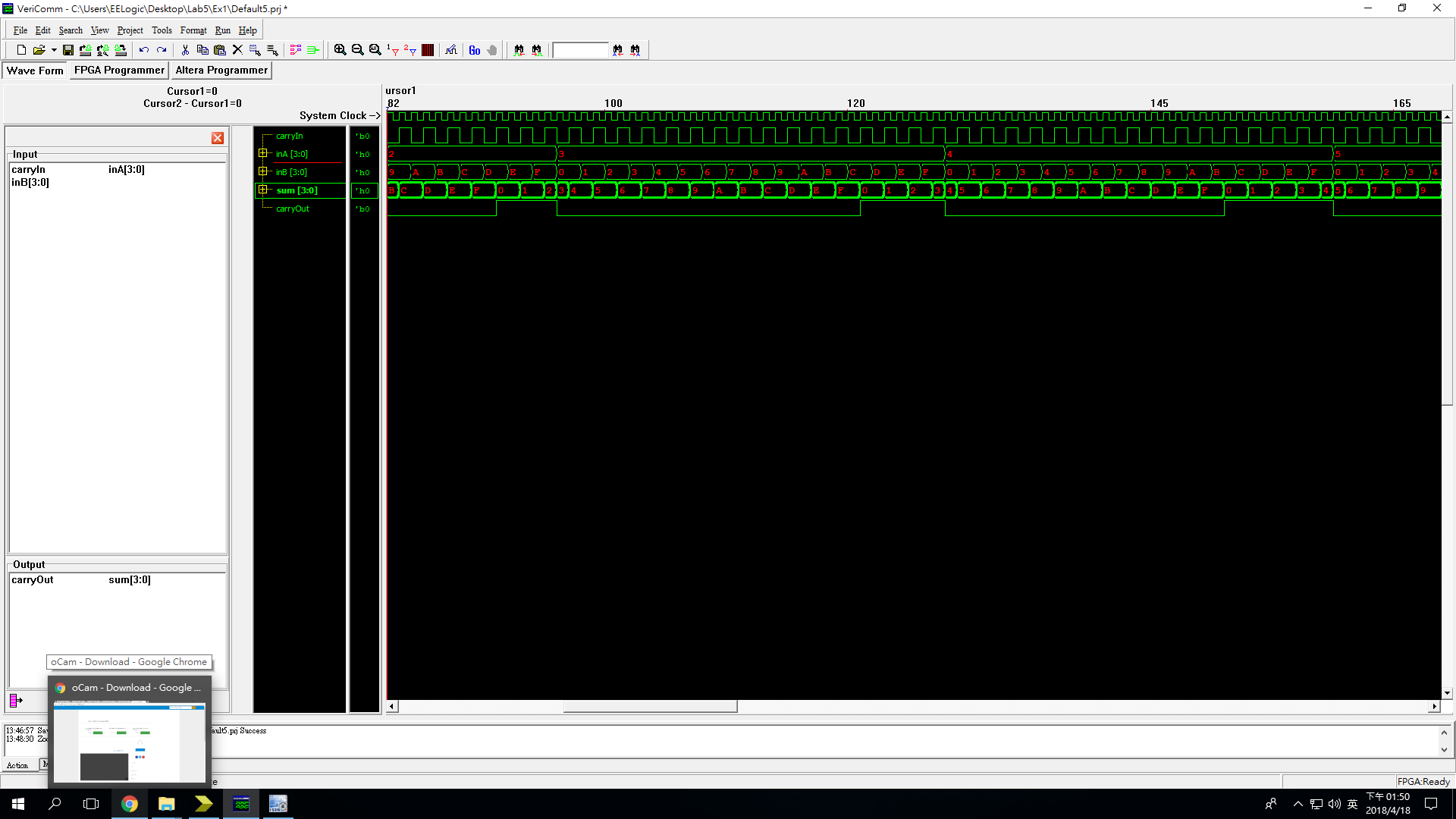
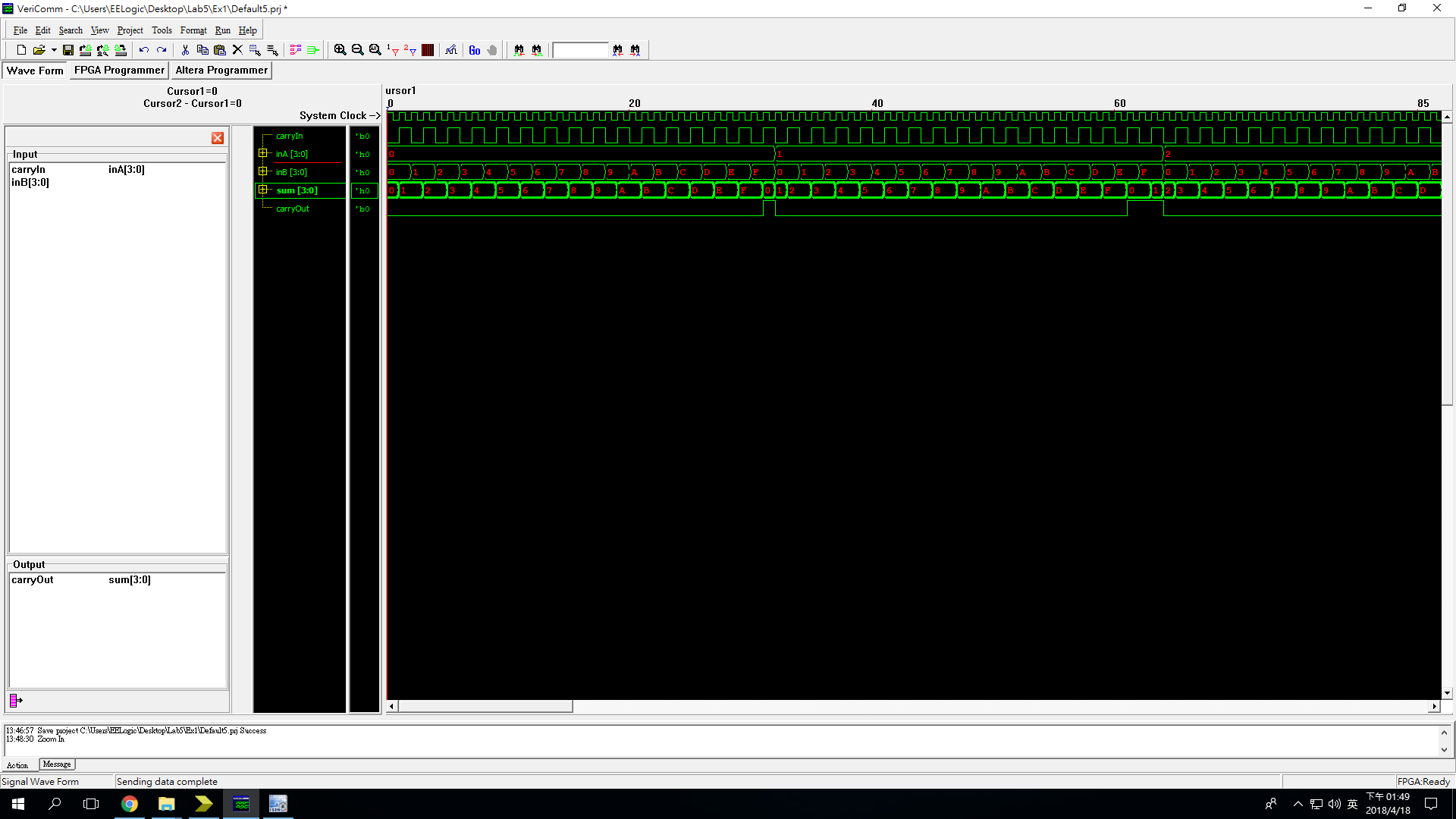
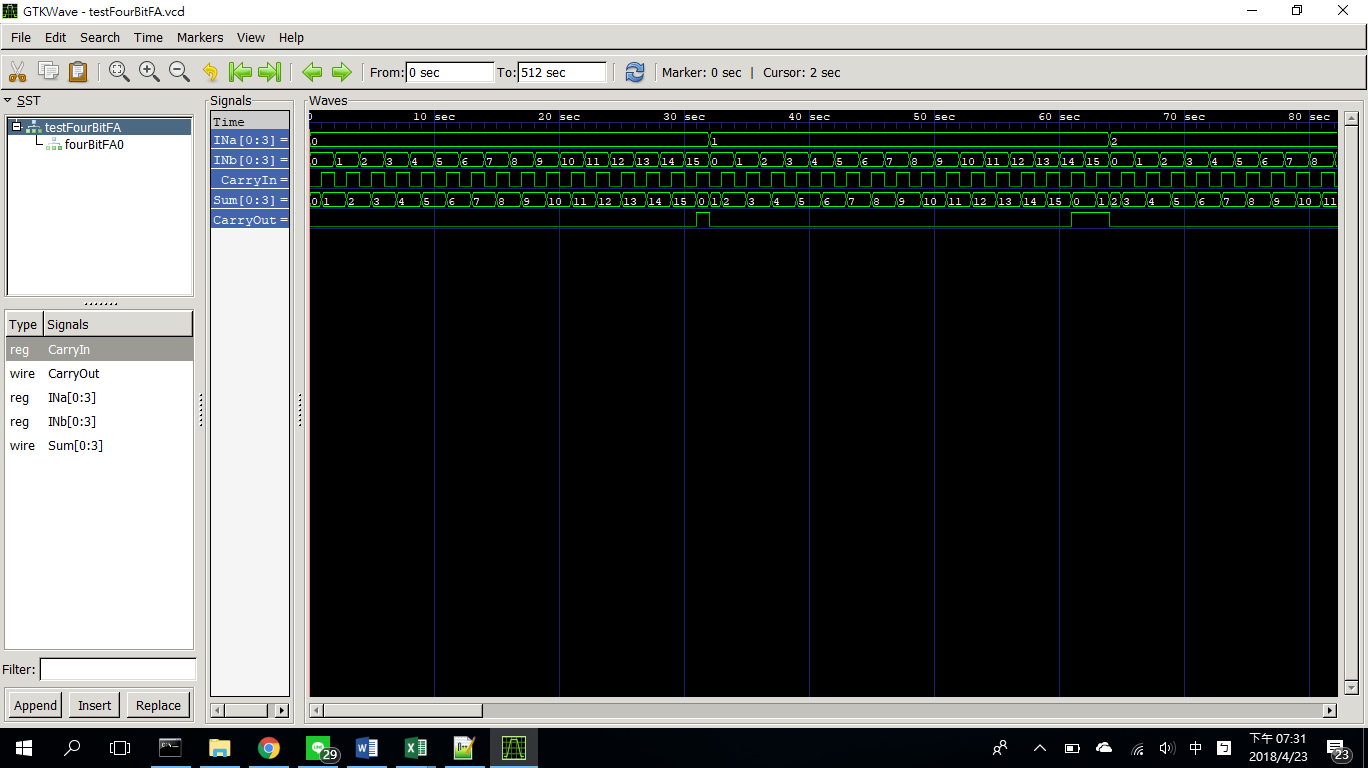
CarryIn=0, INa=0001, INb=0000, Sum=0001, CarryOut=0

CarryIn=1, INa=0001, INb=0000, Sum=0010, CarryOut=0

CarryIn=0, INa=0001, INb=0001, Sum=0010, CarryOut=0

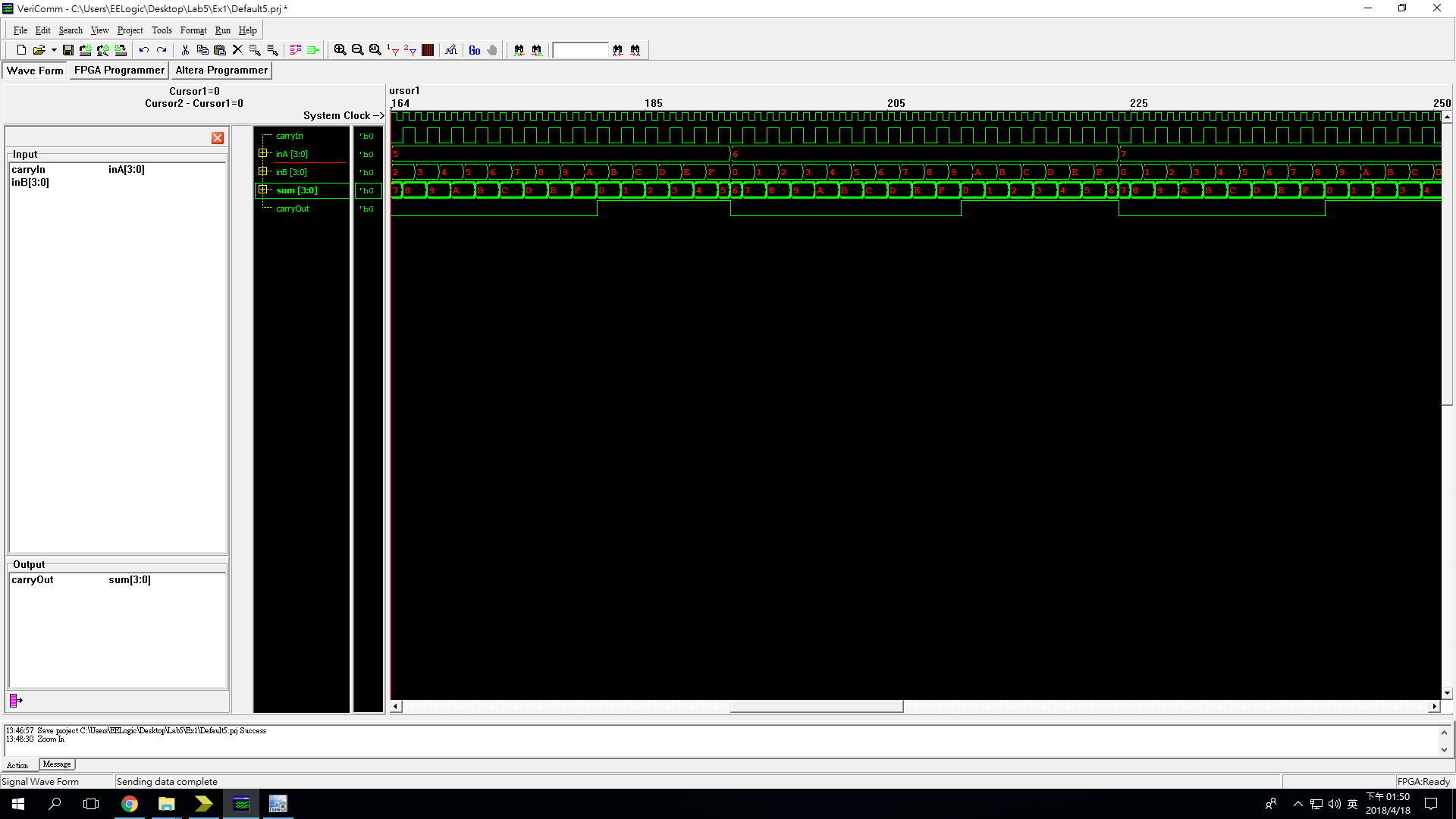
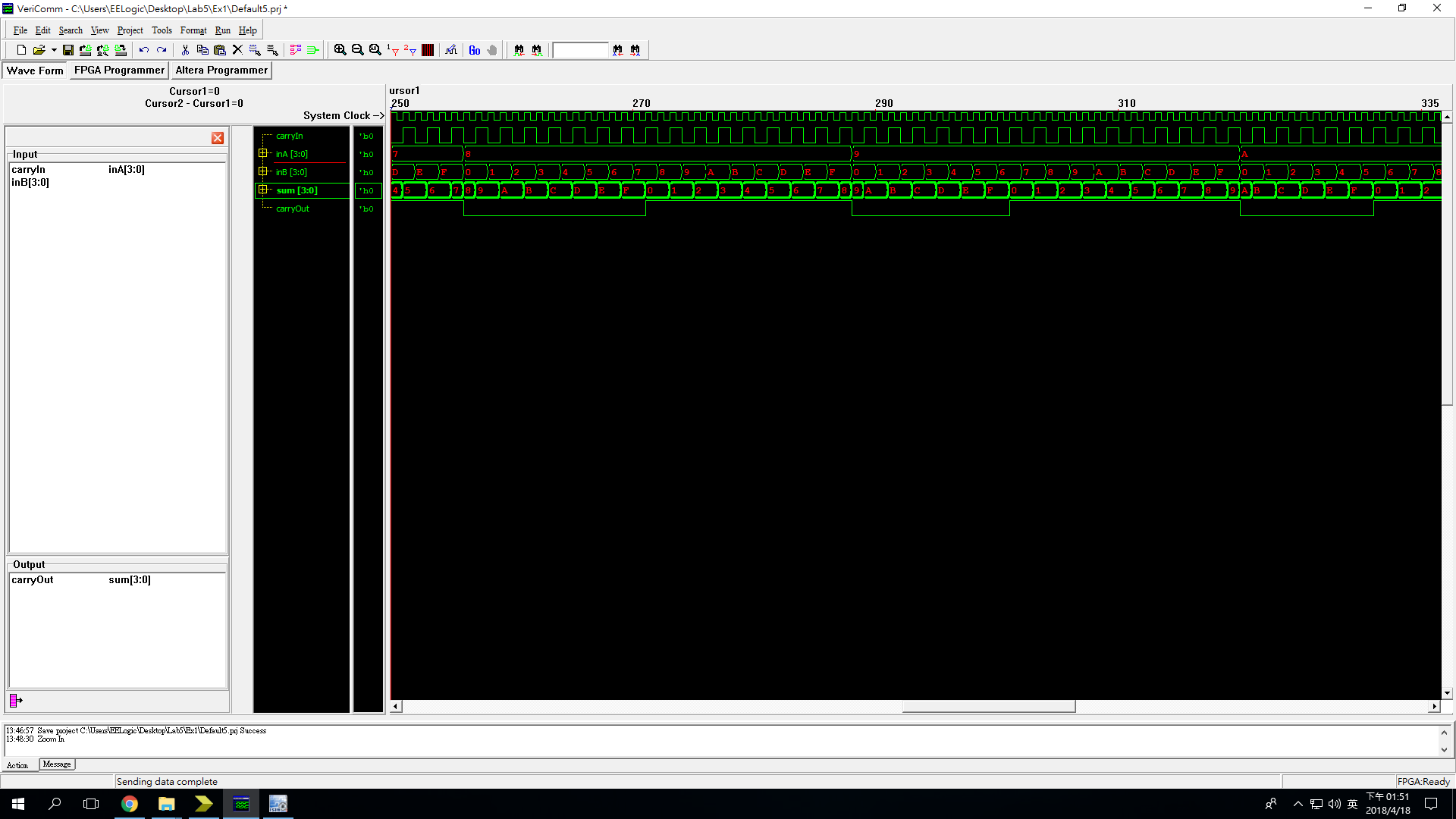
CarryIn=1, INa=0001, INb=0001, Sum=0011, CarryOut=0

CarryIn=0, INa=0001, INb=0010, Sum=0011, CarryOut=0

(4 bit 加法器模擬後的波形圖-節錄)

(4 bit 加法器驗證後的波形圖-1)

(4 bit 加法器驗證後的波形圖-2)

(4 bit 加法器驗證後的波形圖-3)

(4 bit 加法器驗證後的波形圖-4)

4 bit 加法器在資料流層次的verilog 只需要單純地將其相加，因此編寫較簡單，另外，在模擬以及驗證時也只需確認A+B是否等於{CarryOut, sum}，因此不算太複雜。

1. 基礎題(二)：4bit 算術邏輯單元：

在此實驗中，我們利用資料處理層次語法寫出4 bit簡易算術邏輯單 元，在模擬器上模擬後，再燒錄至FPGA進行驗證。

以下是verilog code、testbench、模擬後的結果以及驗證結果。

module fourBitALU(in0, in1, sel, out);

input [3:0]in0, in1;

input [2:0]sel;

output[3:0]out;

assign out =

sel[2]?(sel[1]?(sel[0]?in0>in1:in0<<in1):(sel[0]?in0>>in1:in0^in1)):(sel[1]?(sel [0]?in0|in1:in0&in1):(sel[0]?in0-in1:in0+in1));

endmodule

(4 bit 簡易算術邏輯單元的verilog code)

(4 bit 簡易算術邏輯單元的test bench)

module fourBitALUTest;

reg [3:0]in0, in1;

reg [2:0]sel;

reg [4:0]i;

wire[3:0]out;

fourBitALU ALU0(.in0(in0), .in1(in1), .sel(sel), .out(out));

initial begin

in0=4'b1100;

in1=4'b0001;

sel=3'd0;

for(i=5'd0; i<7; i=i+1)begin

#1 sel=sel+1;

end

#1;

in0=4'b0001;

in1=4'b1101;

sel=3'd0;

for(i=5'd0; i<7; i=i+1)begin

#1 sel=sel+1;

end

end

endmodule

(4 bit 算術邏輯單元模擬結果)

in0=1100, in1=0001, out=1101, sel=000

in0=1100, in1=0001, out=1011, sel=001

in0=1100, in1=0001, out=0000, sel=010

in0=1100, in1=0001, out=1101, sel=011

in0=1100, in1=0001, out=1101, sel=100

in0=1100, in1=0001, out=0110, sel=101

in0=1100, in1=0001, out=1000, sel=110

in0=1100, in1=0001, out=0001, sel=111

in0=0001, in1=1101, out=1110, sel=000

in0=0001, in1=1101, out=0100, sel=001

in0=0001, in1=1101, out=0001, sel=010

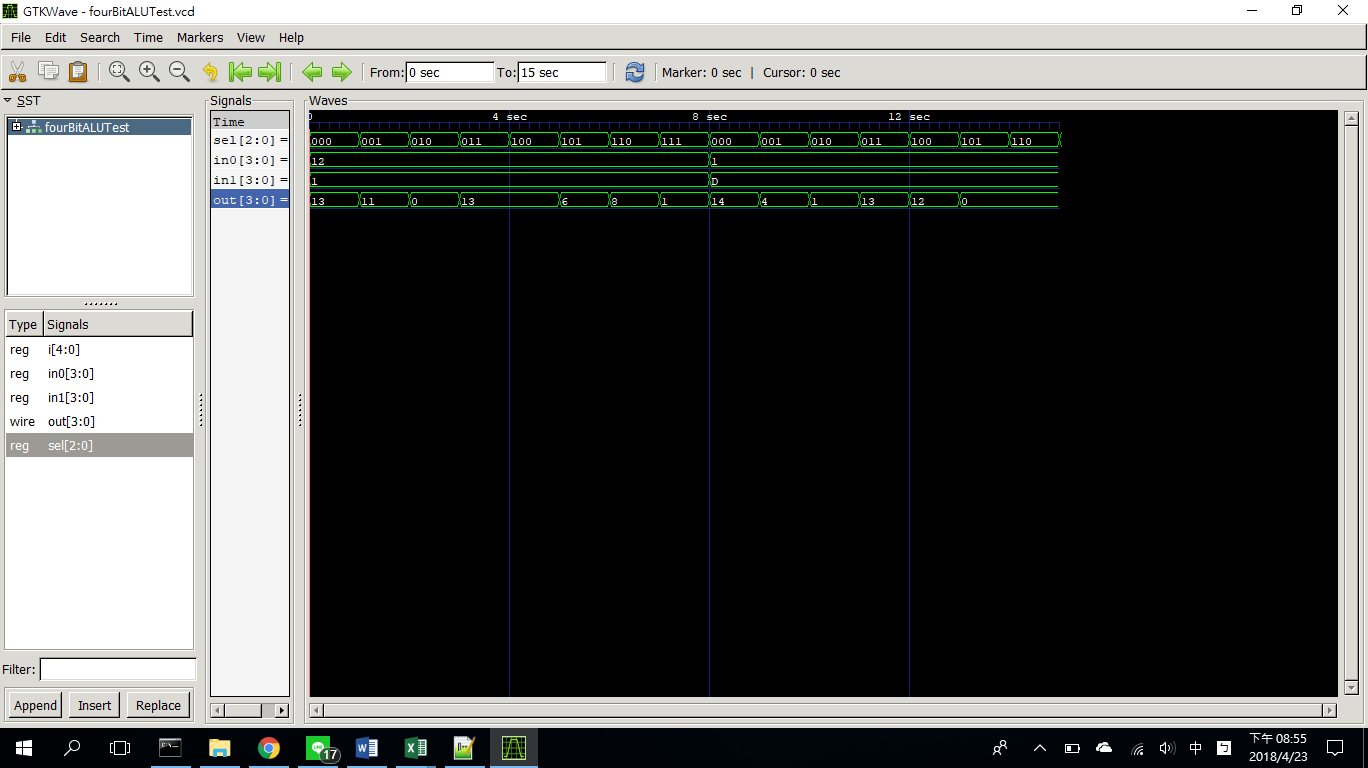
in0=0001, in1=1101, out=1101, sel=011

in0=0001, in1=1101, out=1100, sel=100

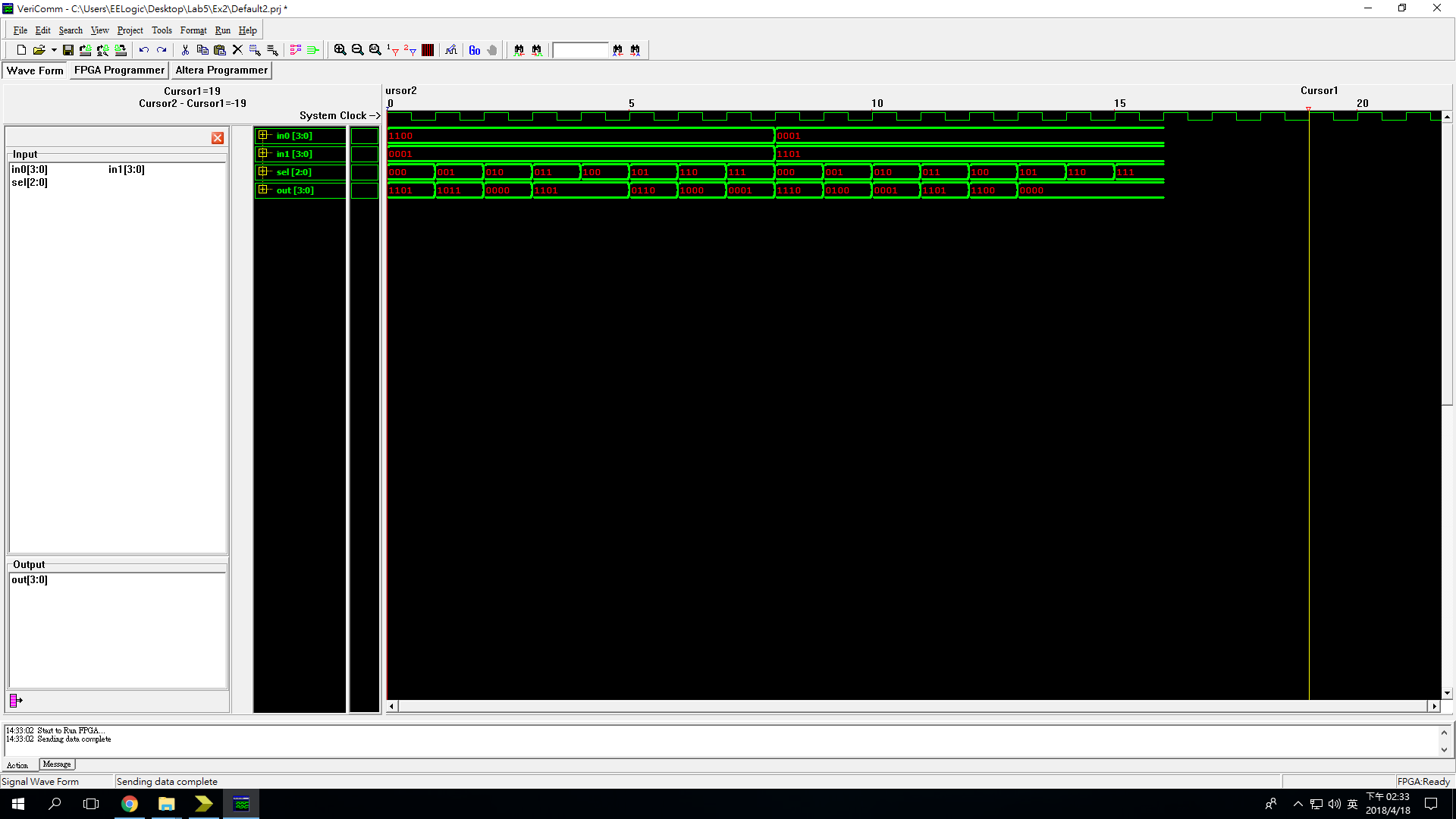
in0=0001, in1=1101, out=0000, sel=101

in0=0001, in1=1101, out=0000, sel=110

in0=0001, in1=1101, out=0000, sel=111



(4 bit 算術邏輯單元模擬波形圖)



(4 bit 算術邏輯單元驗證波形圖)

算術邏輯單元有點類似先將各式結果先算好，再接上多工器(以這次實 驗的例子就是8對1多工器)，以輸出所需的結果。

1. 挑戰題：4bit 前瞻式加法器：

在此實驗中，我們利用資料處理層次語法寫出4 bit前瞻式加法器，在 模擬器上模擬後，再燒錄至FPGA進行驗證。

以下是verilog code、testbench、模擬後的結果以及驗證結果。

module fourBitAheadAdder(inA, inB, carryIn, sum, carryOut);

input [3:0]inA, inB;

input carryIn;

output [3:0]sum;

output carryOut;

assign {carryOut, sum}

={((inA[3]&inB[3])|(inA[3]^inB[3])&((inA[2]&inB[2])|((inA[2]^inB[2])&((inA[1]&inB[1])|((inA[1]^inB[1])&((inA[0]&inB[0])|((inA[0]^inB[0])&carryIn)))))))

,inA[3]^inB[3]^((inA[2]&inB[2])|((inA[2]^inB[2])&((inA[1]&inB[1])|((inA[1]^inB[1])&((inA[0]&inB[0])|((inA[0]^inB[0])&carryIn))))))

,inA[2]^inB[2]^((inA[1]&inB[1])|((inA[1]^inB[1])&((inA[0]&inB[0])|((inA[0]^inB[0])&carryIn))))

, inA[1]^inB[1]^((inA[0]&inB[0])|((inA[0]^inB[0])&carryIn))

, inA[0]^inB[0]^carryIn};

endmodule

(4bit 前瞻式加法器 Verilog code)

(4 bit 前瞻式加法器 test bench)

module fourBitLACTest;

reg [3:0]inA, inB;

reg carryIn;

wire [3:0]sum;

wire carryOut;

fourBitAheadAdder FA0(inA, inB, carryIn, sum, 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

endmodule

(4 bit 前瞻式加法器 模擬結果-節錄)

carryIn=1, inA=1010, inB=1111, Sum=1010, CarryOut=1

carryIn=0, inA=1011, inB=0000, Sum=1011, CarryOut=0

carryIn=1, inA=1011, inB=0000, Sum=1100, CarryOut=0

carryIn=0, inA=1011, inB=0001, Sum=1100, CarryOut=0

carryIn=1, inA=1011, inB=0001, Sum=1101, CarryOut=0

carryIn=0, inA=1011, inB=0010, Sum=1101, CarryOut=0

carryIn=1, inA=1011, inB=0010, Sum=1110, CarryOut=0

carryIn=0, inA=1011, inB=0011, Sum=1110, CarryOut=0

carryIn=1, inA=1011, inB=0011, Sum=1111, CarryOut=0

carryIn=0, inA=1011, inB=0100, Sum=1111, CarryOut=0

carryIn=1, inA=1011, inB=0100, Sum=0000, CarryOut=1

carryIn=0, inA=1011, inB=0101, Sum=0000, CarryOut=1

carryIn=1, inA=1011, inB=0101, Sum=0001, CarryOut=1

carryIn=0, inA=1011, inB=0110, Sum=0001, CarryOut=1

carryIn=1, inA=1011, inB=0110, Sum=0010, CarryOut=1

carryIn=0, inA=1011, inB=0111, Sum=0010, CarryOut=1

carryIn=1, inA=1011, inB=0111, Sum=0011, CarryOut=1

carryIn=0, inA=1011, inB=1000, Sum=0011, CarryOut=1

carryIn=1, inA=1011, inB=1000, Sum=0100, CarryOut=1

carryIn=0, inA=1011, inB=1001, Sum=0100, CarryOut=1

carryIn=1, inA=1011, inB=1001, Sum=0101, CarryOut=1

carryIn=0, inA=1011, inB=1010, Sum=0101, CarryOut=1

carryIn=1, inA=1011, inB=1010, Sum=0110, CarryOut=1

carryIn=0, inA=1011, inB=1011, Sum=0110, CarryOut=1

carryIn=1, inA=1011, inB=1011, Sum=0111, CarryOut=1

carryIn=0, inA=1011, inB=1100, Sum=0111, CarryOut=1

carryIn=1, inA=1011, inB=1100, Sum=1000, CarryOut=1

carryIn=0, inA=1011, inB=1101, Sum=1000, CarryOut=1

carryIn=1, inA=1011, inB=1101, Sum=1001, CarryOut=1

carryIn=0, inA=1011, inB=1110, Sum=1001, CarryOut=1

carryIn=1, inA=1011, inB=1110, Sum=1010, CarryOut=1

carryIn=0, inA=1011, inB=1111, Sum=1010, CarryOut=1

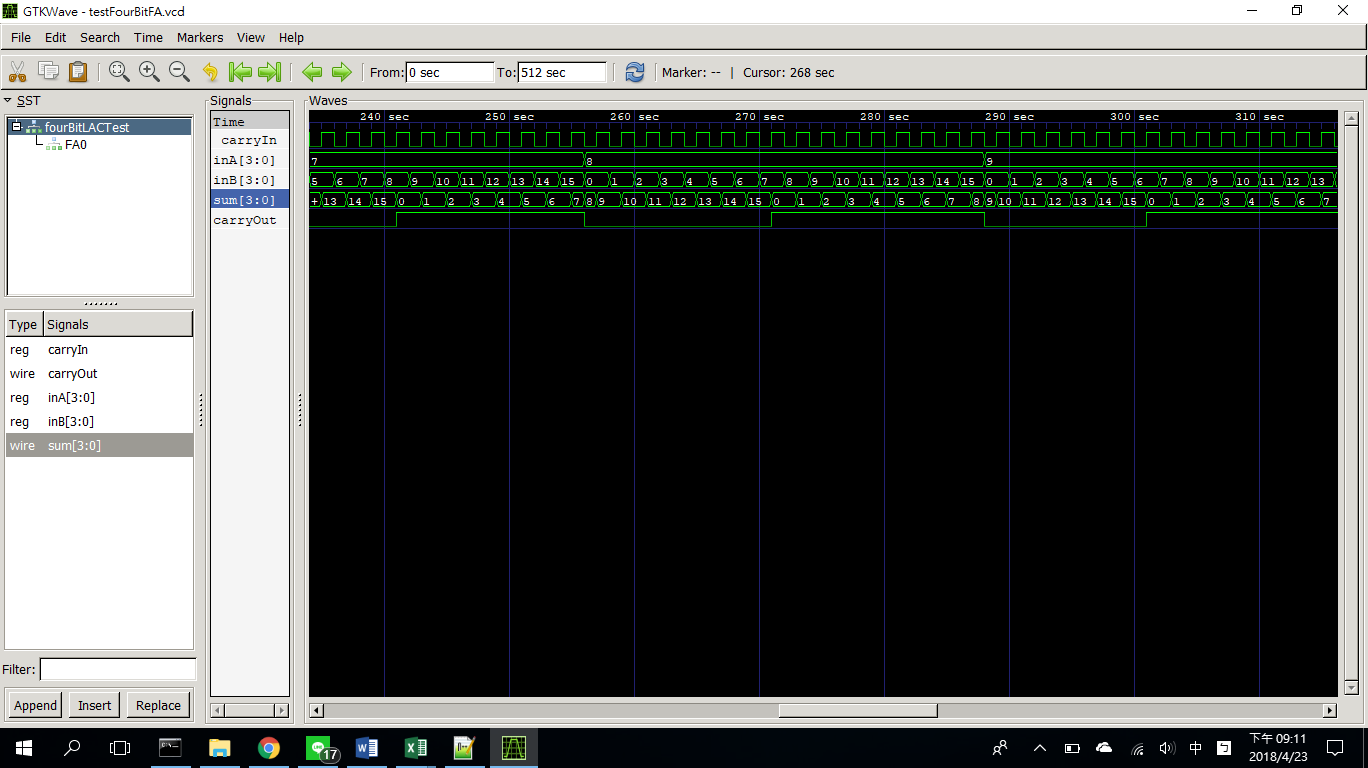
carryIn=1, inA=1011, inB=1111, Sum=1011, CarryOut=1

carryIn=0, inA=1100, inB=0000, Sum=1100, CarryOut=0

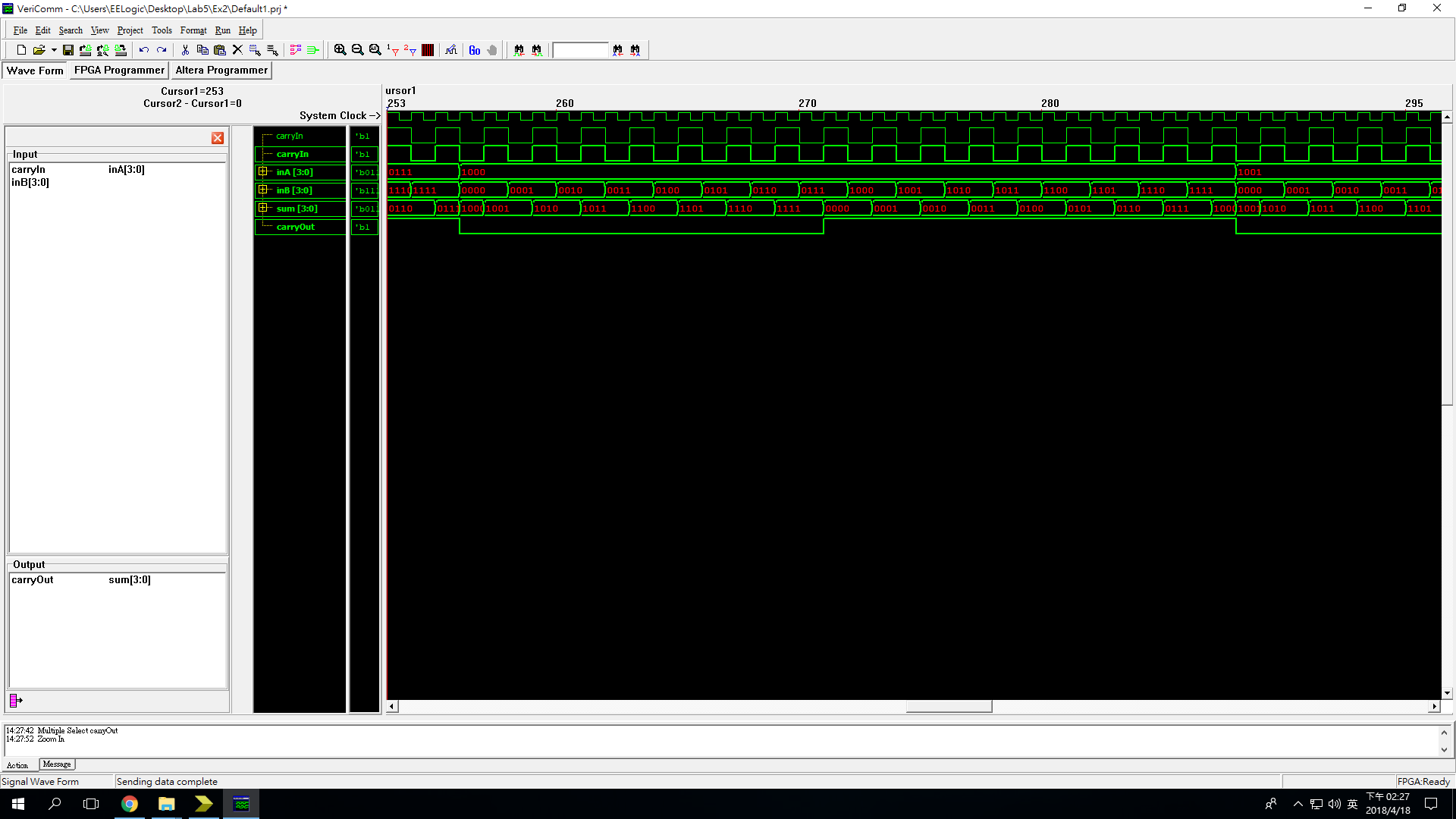
carryIn=1, inA=1100, inB=0000, Sum=1101, CarryOut=0

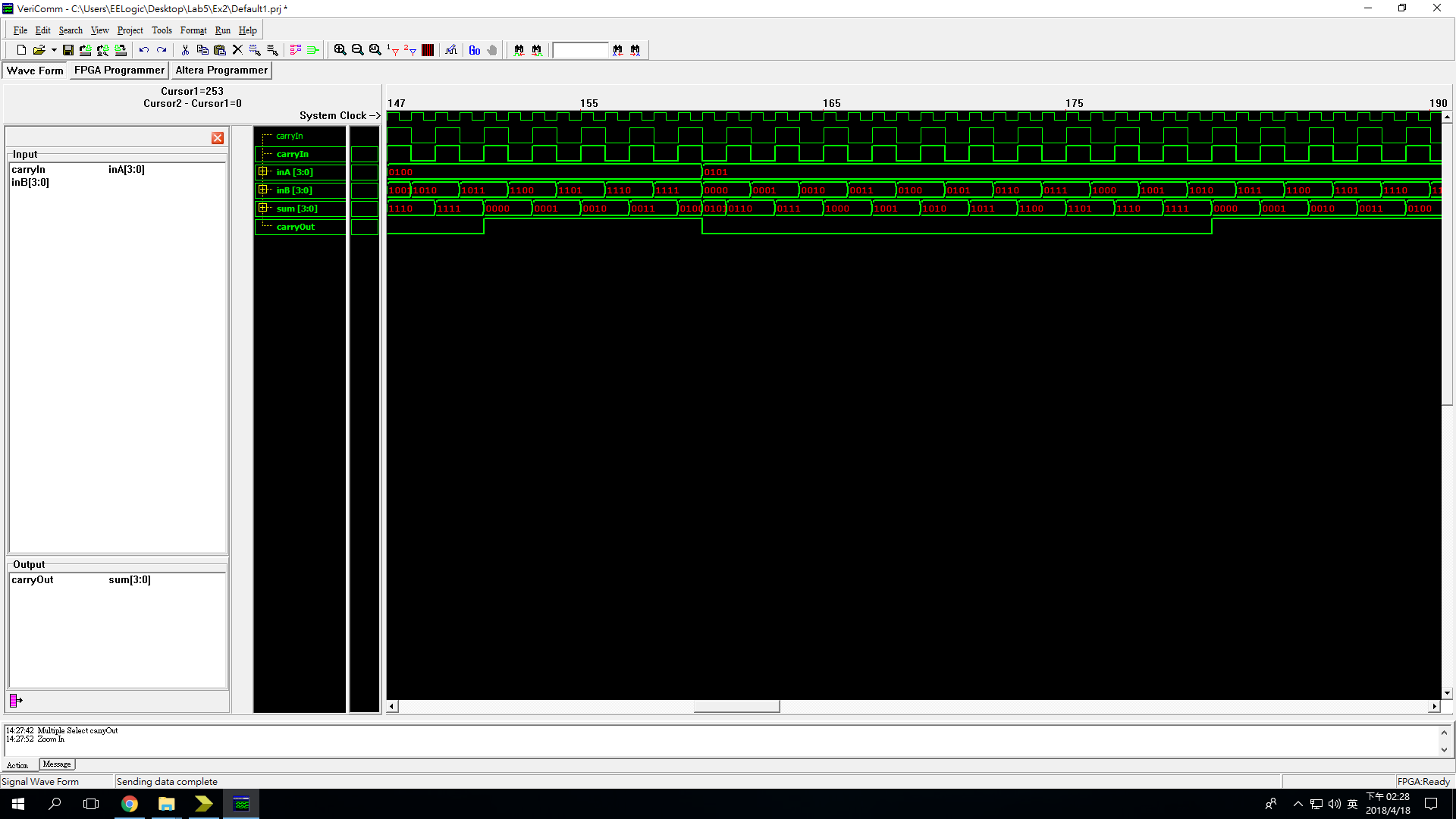
carryIn=0, inA=1100, inB=0001, Sum=1101, CarryOut=0

carryIn=1, inA=1100, inB=0001, Sum=1110, CarryOut=0



(4 bit 前瞻式加法器模擬波形圖-節錄)



(4 bit 前瞻式加法器驗證波形圖-1)

(4 bit 前瞻式加法器驗證波形圖-2)

前瞻式加法器的概念是直接以input來計算每個位數的進位，而非等 待前一位元的進位來計算下一位的進位，如此一來可以省去等待前方位原 傳導的時間，因此速度較快。

心得：

1. 張軒：

隨著時間的過去，final project的日子越來越近了，雖然完成了lab5但是感覺還是什麼都不會呀...(不過總算搞懂什麼時候要用wire什麼時候要用reg了)。第一次學到"ARM"是小學的時候老師說那是手臂，第二次則是在大學結果實驗趴ㄋ和助教跟我說那是cpu(what?!...)。應該不只有我不知道原來手臂還有這麼深澳的衍伸意涵...吧?不過看起來lab6~lab9還有很多verilog的資料，希望在那之後我的手臂可以又粗又壯。在各種意義上都是~

1. 章子嚴：

Lab 5我已經不是很懂要這麼寫，我不是很了解什麼時候要名字要大小寫，我感覺有點跟不上進度，雖然我懂組員大概是在寫什麼但是就有些部分就不是很懂。我看著組員寫的時候不懂他們寫什麼，請教他們后才懂。原來有不同的寫法像inC , inC(.INc)，原本是寫inC程式有錯誤，寫成inC(.INc)就可以跑了真奇怪。

1. 魏晉成：

在Lab5中，看到自己熟悉的運算子+, -, \*, /, &等等，就感覺安心很 多，有種自己在寫C的感覺，再加上之後可以用switch case還有if-else， 程式碼就變得更簡潔，也不必每每把2對1多工器層層堆疊了。

不過寫到這裡我就頗為好奇，到底該如何把這種接近高階語言的東西 轉換成Gate Level的表示，也許可以把「寫出Verilog 編譯器」列為以後的 代辦事項。