

**DIGITAL DESIGN**

**ASSIGNMENT REPORT**

**ASSIGNMENT ID : 1**

**Student Name: 刘乐奇**

**Student ID: 12011327**

PART 1: DIGITAL design THEORY

Provide your answers here:

1. (a) 1 6384 bytes

(b) 3355 4432 bytes

(c) 34 3597 3836.8 bytes

2. Without sign: In binary: (1111 1111 1111)2

In decimal: (4095)10

In hexadecimal: (fff)16

With sign: In binary: (0111 1111 1111)2

In decimal: (2047)10

In hexadecimal: (7ff)16

3. (a) (248)10 = (1111 1000)2

(b) (248)10 = (f8)16 = (1111 1000)2

I think method (b) may be faster. Because when we divide the same number but by a larger number, it will cost less time. In the meanwhile, when convert hexadecimal to binary, we can convert 4-bit a time.

4. a. 9’s complement: (747269063)10

10’s complement: (747269064)10

b. 9’s complement: (35677389)10

10’s complement: (35677390)10

5. (a) (3941)16

(b) (1100 0110 1011 1111)2

(c) (0011 1001 0100 0001)2

(d) (3941)16 which is the same as the result in (a)

6. (a) (10111.1010111111001)2

(b) (5/3)10 = (1.10101010)2 with 8 precisions

(1.10101010)2 = (1.6640625)10 which has the difference (1.6666666)10 - (1.6640625)10 = (0.0026041)10

(c) (1.10101010)2 = (1.aa)16 = (1.6640625)10 whose result is the same as directly converted to decimal. Because hexadecimal is equivalent as binary in some degree. When converting binary to hexadecimal, we usually do conversion from every 4 bits in binary to 1 bit in hexadecimal, vice versa.

7. (a) (975)BCD

(b) (642)excess-3

(c) (713)84-2-1

(d) (754)6311

(e) (1001 0111 0101)2 = (2421)10

8. (a) A AND B = (0100 1010)2 = (4a)16

(b) A OR B = (1101 1110)2 = (de)16

(c) A XOR B = (1001 0100)2 = (94)16

(d) NOT A = (0010 0101)2 = (25)16

(e) NOT B = (1011 0001)2 = (b1)16

(f) A NAND B = (1011 0101)2 = (b5)16

(g) A NOR B = (0010 0001)2 = (21)16

PART 2: DIGITAL design LAB (Task1)

##### Design

*Describe the design of your system by providing the following information:*

* ***Verilog design (provide the Verilog code)***

*//UnisignedAddition.v*

*`timescale 1ns / 1ps*

*module UnsignedAddition(in1,in2,out);*

*parameter WIDTH=2;*

*input [WIDTH-1:0] in1;*

*input [WIDTH-1:0] in2;*

*output [WIDTH:0] out;*

*wire [WIDTH:0] c1, c2, tmp1, tmp2;*

*assign tmp1 = in1^in2;*

*assign c1 = in1&in2;*

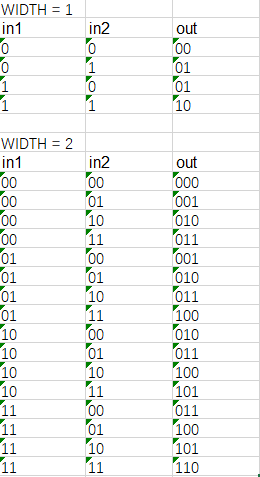
*assign tmp2 = tmp1^(c1<<1);*

*assign c2 = tmp1&(c1<<1);*

*assign out = tmp2^(c2<<1);*

*endmodule*

* ***Truth-table***



##### simulation

*Describe how you build the test bench and do the simulation.*

* ***Using Verilog (provide the Verilog code)***

*//UnisignedAddition\_sim.v*

*`timescale 1ns / 1ps*

*module UnsignedAddition\_sim();*

*reg [0:0] in1\_1\_sim,in2\_1\_sim;*

*wire [1:0] out\_1\_sim;*

*reg [1:0] in1\_2\_sim,in2\_2\_sim;*

*wire [2:0] out\_2\_sim;*

*UnsignedAddition #(2) ub(*

*.in1(in1\_2\_sim),*

*.in2(in2\_2\_sim),*

*.out(out\_2\_sim)*

*);*

*UnsignedAddition #(1) ua(*

*.in1(in1\_1\_sim),*

*.in2(in2\_1\_sim),*

*.out(out\_1\_sim)*

*);*

*initial begin*

*in1\_2\_sim = 2'b00; in2\_2\_sim = 2'b00; in1\_1\_sim = 1'b0; in2\_1\_sim = 1'b0;*

*#10*

*in1\_2\_sim = 2'b00; in2\_2\_sim = 2'b01; in1\_1\_sim = 1'b0; in2\_1\_sim = 1'b1;*

*#10*

*in1\_2\_sim = 2'b00; in2\_2\_sim = 2'b10; in1\_1\_sim = 1'b1; in2\_1\_sim = 1'b0;*

*#10*

*in1\_2\_sim = 2'b00; in2\_2\_sim = 2'b11; in1\_1\_sim = 1'b1; in2\_1\_sim = 1'b1;*

*#10*

*in1\_2\_sim = 2'b01; in2\_2\_sim = 2'b00;*

*#10*

*in1\_2\_sim = 2'b01; in2\_2\_sim = 2'b01;*

*#10*

*in1\_2\_sim = 2'b01; in2\_2\_sim = 2'b10;*

*#10*

*in1\_2\_sim = 2'b01; in2\_2\_sim = 2'b11;*

*#10*

*in1\_2\_sim = 2'b10; in2\_2\_sim = 2'b00;*

*#10*

*in1\_2\_sim = 2'b10; in2\_2\_sim = 2'b01;*

*#10*

*in1\_2\_sim = 2'b10; in2\_2\_sim = 2'b10;*

*#10*

*in1\_2\_sim = 2'b10; in2\_2\_sim = 2'b11;*

*#10*

*in1\_2\_sim = 2'b11; in2\_2\_sim = 2'b00;*

*#10*

*in1\_2\_sim = 2'b11; in2\_2\_sim = 2'b01;*

*#10*

*in1\_2\_sim = 2'b11; in2\_2\_sim = 2'b10;*

*#10*

*in1\_2\_sim = 2'b11; in2\_2\_sim = 2'b11;*

*#10*

*$finish();*

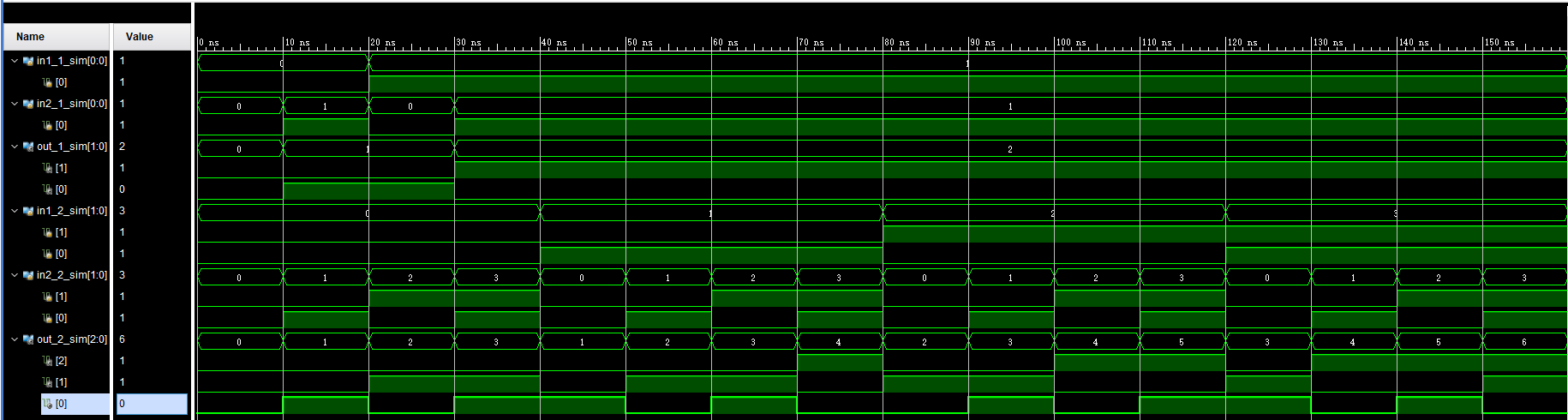
*end*

*endmodule*

* ***Wave form of simulation result (provide screen shots)***

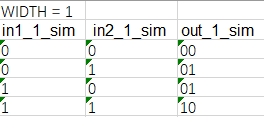
For width = 1, input in1\_1\_sim, in2\_1\_sim, ouput out\_1\_sim;

For width = 2, input in1\_2\_sim, in2\_2\_sim, ouput out\_2\_sim;

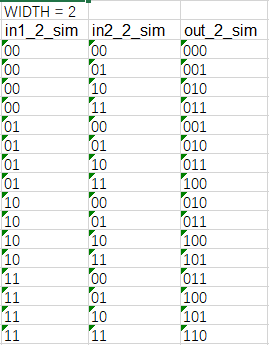


***The description on whether the simulation result is same as the truth-table, is the function of the design meet the expectation.***

We can know from the waveform that when the width is 1, the result is in the below:



When the width is 2, the result is in the below:



Comparing the above results with truth-table, the simulation result is well matched. The function I designed well meets the expectation.

##### the description of operation

*Describe the problem occurred while in the lab and your solution. Any suggestions are welcomed.*

* ***Problems and solutions***

(1) Different with other IDE such as IDEA, pycharm, etc., the error in Vivado is more hard to find. Once I wrote “Endmodule” instead of “endmodule”, I could not find that though Vivado had warned me because of the obscure indicator.

(2) Initially, I had used “+” for addition. However, I thought it might be simple so that I changed my code to current one, which can hold on at most 2-bit unsigned addition by using bitwise logic operation.

(3) To simplify my code, I used parameter WIDTH to control the width of input.

PART 2: DIGITAL design LAB (Task2)

##### Design

*Describe the design of your system by providing the following information:*

* ***Verilog design while using data flow (provide the Verilog code)***

*//distributive1bit\_df.v*

*`timescale 1ns / 1ps*

*module distributive1bit\_df(a,b,c,outa1,outa2,outb1,outb2);*

*input a,b,c;*

*output outa1,outa2, outb1,outb2;*

*///to verify (a)*

*wire xa,ya,za;*

*assign xa = b|c;*

*assign outa1 = a&xa;*

*assign ya = a&b;*

*assign za = a&c;*

*assign outa2 = ya|za;*

*//to verify (b)*

*wire xb,yb,zb;*

*assign xb = b&c;*

*assign outb1 = a|xb;*

*assign yb = a|b;*

*assign zb = a|c;*

*assign outb2 = yb&zb;*

*endmodule*

*//distributive2bit\_df.v*

*`timescale 1ns / 1ps*

*module distributive2bit\_df(a,b,c,outa1,outa2,outb1,outb2);*

*input [1:0] a,b,c;*

*output [1:0] outa1,outa2,outb1,outb2;*

*//to verify (a)*

*wire [1:0] xa,ya,za;*

*assign xa = b|c;*

*assign outa1 = a&xa;*

*assign ya = a&b;*

*assign za = a&c;*

*assign outa2 = ya|za;*

*//to verify (b)*

*wire [1:0] xb,yb,zb;*

*assign xb = b&c;*

*assign outb1 = a|xb;*

*assign yb = a|b;*

*assign zb = a|c;*

*assign outb2 = yb&zb;*

*endmodule*

* ***Verilog design while using structured design (provide the Verilog code)***

*//distributive1bit\_sd.v*

*`timescale 1ns / 1ps*

*module distributive1bit\_sd(a,b,c,outa1,outa2,outb1,outb2);*

*input a,b,c;*

*output outa1,outa2,outb1,outb2;*

*//to verify (a)*

*wire xa,ya,za;*

*or ora\_1(xa,b,c);*

*and anda\_1(outa1,a,xa);*

*and anda\_2(ya,a,b);*

*and anda\_3(za,a,c);*

*or ora\_2(outa2,ya,za);*

*//to verify (b)*

*wire xb,yb,zb;*

*and andb\_1(xb,b,c);*

*or orb\_1(outb1,a,xb);*

*or orb\_2(yb,a,b);*

*or orb\_3(zb,a,c);*

*and andb\_2(outb2,yb,zb);*

*endmodule*

*//distributive2bit\_sd.v*

*`timescale 1ns / 1ps*

*module distributive2bit\_sd(a,b,c,outa1,outa2,outb1,outb2);*

*input [1:0] a,b,c;*

*output [1:0] outa1,outa2,outb1,outb2;*

*//to verify (a)*

*wire [1:0] xa,ya,za;*

*or ora\_1\_1(xa[0],b[0],c[0]);*

*or ora\_1\_2(xa[1],b[1],c[1]);*

*and anda\_1\_1(outa1[0],a[0],xa[0]);*

*and anda\_1\_2(outa1[1],a[1],xa[1]);*

*and anda\_2\_1(ya[0],a[0],b[0]);*

*and anda\_2\_2(ya[1],a[1],b[1]);*

*and anda\_3\_1(za[0],a[0],c[0]);*

*and anda\_3\_2(za[1],a[1],c[1]);*

*or ora\_2\_1(outa2[0],ya[0],za[0]);*

*or ora\_2\_2(outa2[1],ya[1],za[1]);*

*//to verify (b)*

*wire [1:0] xb,yb,zb;*

*and andb\_1\_1(xb[0],b[0],c[0]);*

*and andb\_1\_2(xb[1],b[1],c[1]);*

*or orb\_1\_1(outb1[0],a[0],xb[0]);*

*or orb\_1\_2(outb1[1],a[1],xb[1]);*

*or orb\_2\_1(yb[0],a[0],b[0]);*

*or orb\_2\_2(yb[1],a[1],b[1]);*

*or orb\_3\_1(zb[0],a[0],c[0]);*

*or orb\_3\_2(zb[1],a[1],c[1]);*

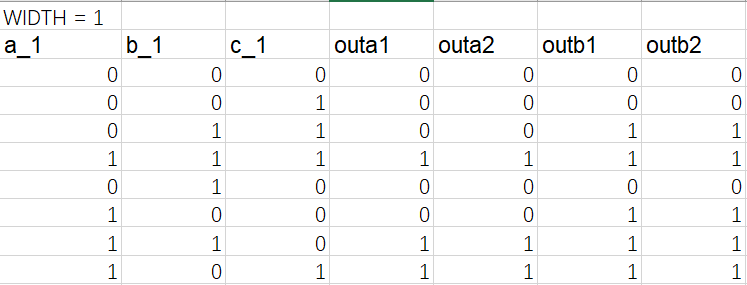
*and andb\_2\_1(outb2[0],yb[0],zb[0]);*

*and andb\_2\_2(outb2[1],yb[1],zb[1]);*

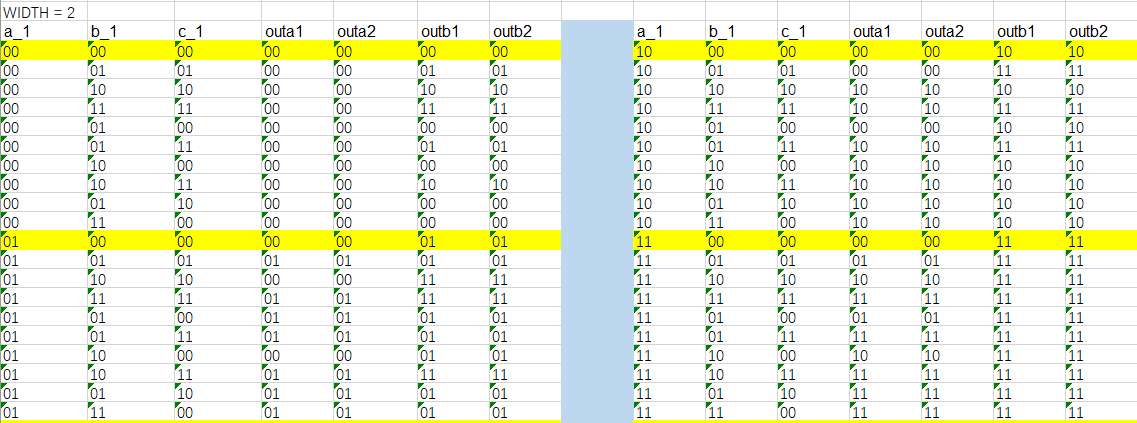
*endmodule*

* ***Truth-table***

*For 1-bit input, outa1 = A(B+C), outa2 = AB+AC, outb1 = A+BC, outb2 = (A+B)(A+C)*



*For 2-bit input, outa1 = A(B+C), outa2 = AB+AC, outb1 = A+BC, outb2 = (A+B)(A+C)*



*It is clearly that outa1 == outa2, outb1 == outb2, no matter whether it is 1-bit or 2-bit.*

##### simulation

*Describe how you build the test bench and do the simulation.*

* ***Using Verilog (provide the Verilog code)***

*//distributive\_sim.v*

*`timescale 1ns / 1ps*

*module distributive\_sim();*

*//test for df-1-bit*

*reg a\_1,b\_1,c\_1;*

*wire outa1\_1\_1,outa2\_1\_1,outb1\_1\_1,outb2\_1\_1;*

*distributive1bit\_df d1(*

*.a(a\_1),*

*.b(b\_1),*

*.c(c\_1),*

*.outa1(outa1\_1\_1),*

*.outa2(outa2\_1\_1),*

*.outb1(outb1\_1\_1),*

*.outb2(outb2\_1\_1)*

*);*

*//test for sd-1-bit*

*wire outa1\_2\_1,outa2\_2\_1,outb1\_2\_1,outb2\_2\_1;*

*distributive1bit\_sd s1(*

*.a(a\_1),*

*.b(b\_1),*

*.c(c\_1),*

*.outa1(outa1\_2\_1),*

*.outa2(outa2\_2\_1),*

*.outb1(outb1\_2\_1),*

*.outb2(outb2\_2\_1)*

*);*

*//test for df-2-bit*

*reg [1:0] a\_2,b\_2,c\_2;*

*wire [1:0] outa1\_1\_2,outa2\_1\_2,outb1\_1\_2,outb2\_1\_2;*

*distributive2bit\_df d2(*

*.a(a\_2),*

*.b(b\_2),*

*.c(c\_2),*

*.outa1(outa1\_1\_2),*

*.outa2(outa2\_1\_2),*

*.outb1(outb1\_1\_2),*

*.outb2(outb2\_1\_2)*

*);*

*//test for sd-2-bit*

*wire [1:0] outa1\_2\_2,outa2\_2\_2,outb1\_2\_2,outb2\_2\_2;*

*distributive2bit\_sd s2(*

*.a(a\_2),*

*.b(b\_2),*

*.c(c\_2),*

*.outa1(outa1\_2\_2),*

*.outa2(outa2\_2\_2),*

*.outb1(outb1\_2\_2),*

*.outb2(outb2\_2\_2)*

*);*

*//test case*

*integer i;*

*initial begin*

*{a\_1,b\_1,c\_1} = 1'b0;*

*for (i=1; i<=7; i=i+1) begin*

*#10 {a\_1,b\_1,c\_1} = {a\_1,b\_1,c\_1} + 1'b1;*

*end*

*#10*

*{a\_1,b\_1,c\_1} = 1'b0;*

*#20*

*{a\_2,b\_2,c\_2} = 2'b00;*

*for (i=1; i<=64; i=i+1) begin*

*#10 {a\_2,b\_2,c\_2} = {a\_2,b\_2,c\_2} + 2'b01;*

*end*

*#20*

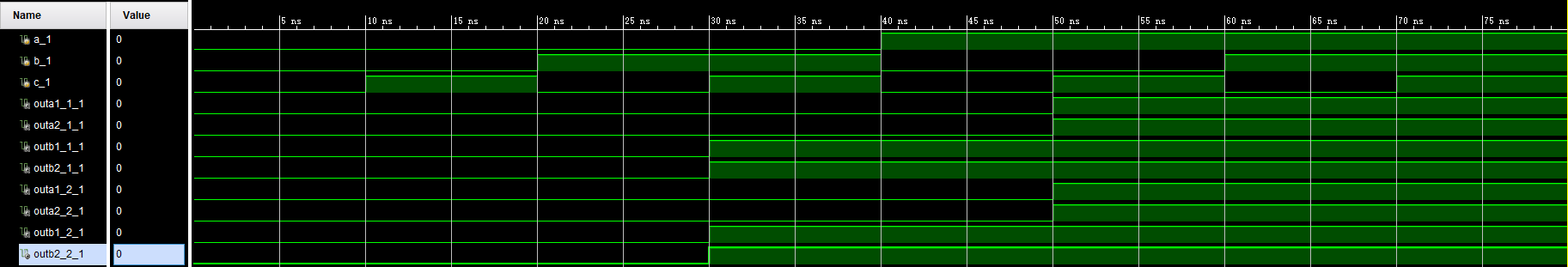
*$finish();*

*End*

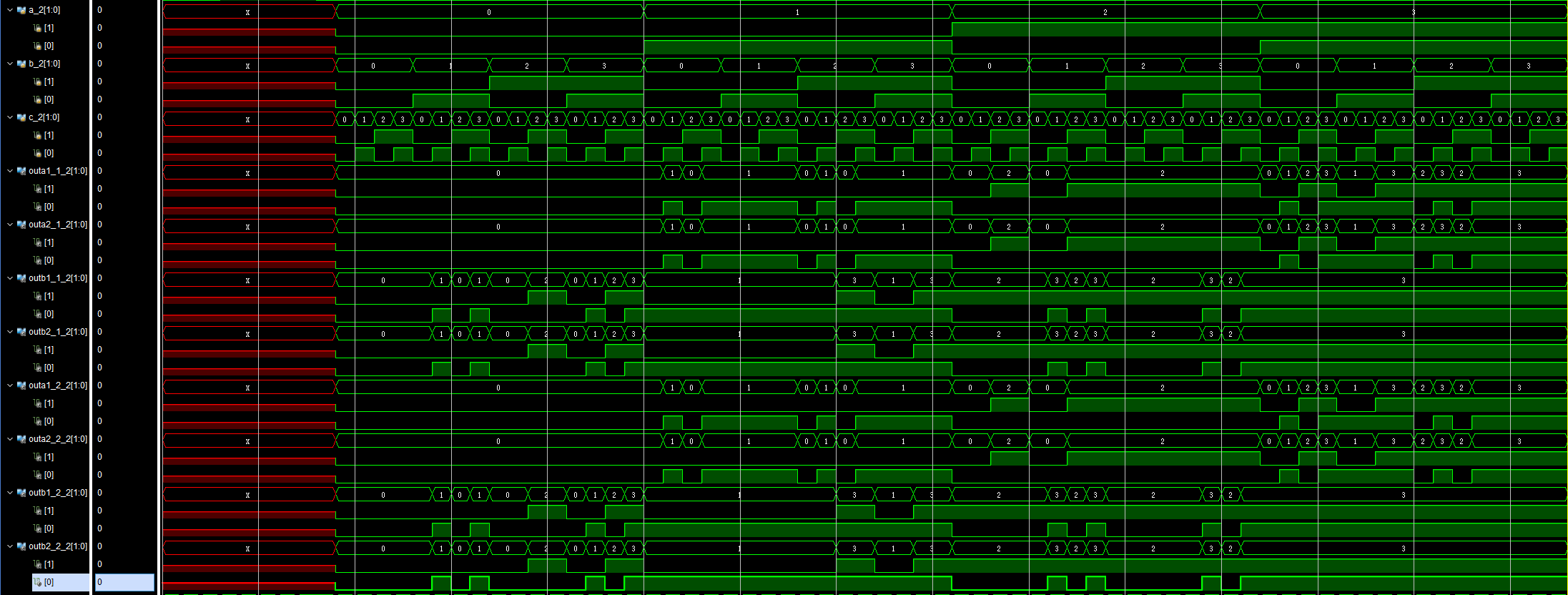
*endmodule*

* ***Wave form of simulation result (provide screen shots)***

*For 1-bit. Start from begin.*

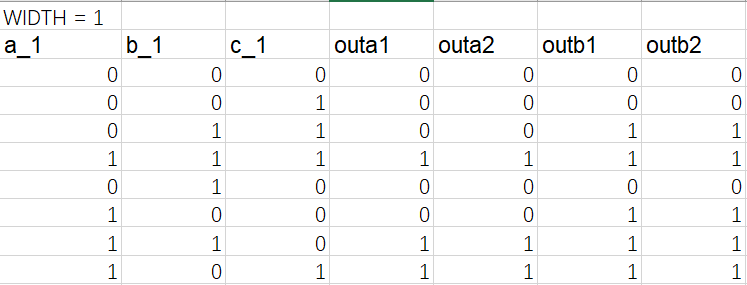


*For 2-bit. Start after 1-bit, that is, after X was over.*

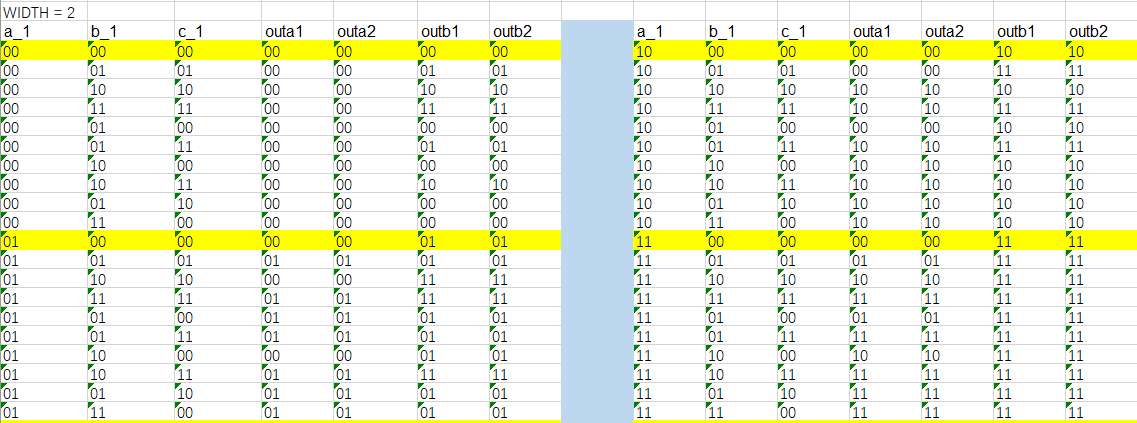


* ***The description on whether the simulation result is same as the truth-table, is the function of the design meet the expectation***

We can know from the waveform that when the width is 1, the result is in the below:



When the width is 2, the result is in the below:



Both two waveforms well meet the truth tables, that is, we can draw a conclusion that the two distributive laws are right and fit no matter whether the inputs are 1-bit or not.

##### the description of operation

*Describe the problem occurred while in the lab and your solution. Any suggestions are welcomed.*

* ***Problems and solutions***

(1) I think it is hard to distinguish the variables when there are plenty of variables. So, naming the variables should be taken as an important thing. Here, I used underline and number to make a difference.

(2) To ensure the simulation can cover all the cases, I used for-loop to assign the value and choose a little larger range.