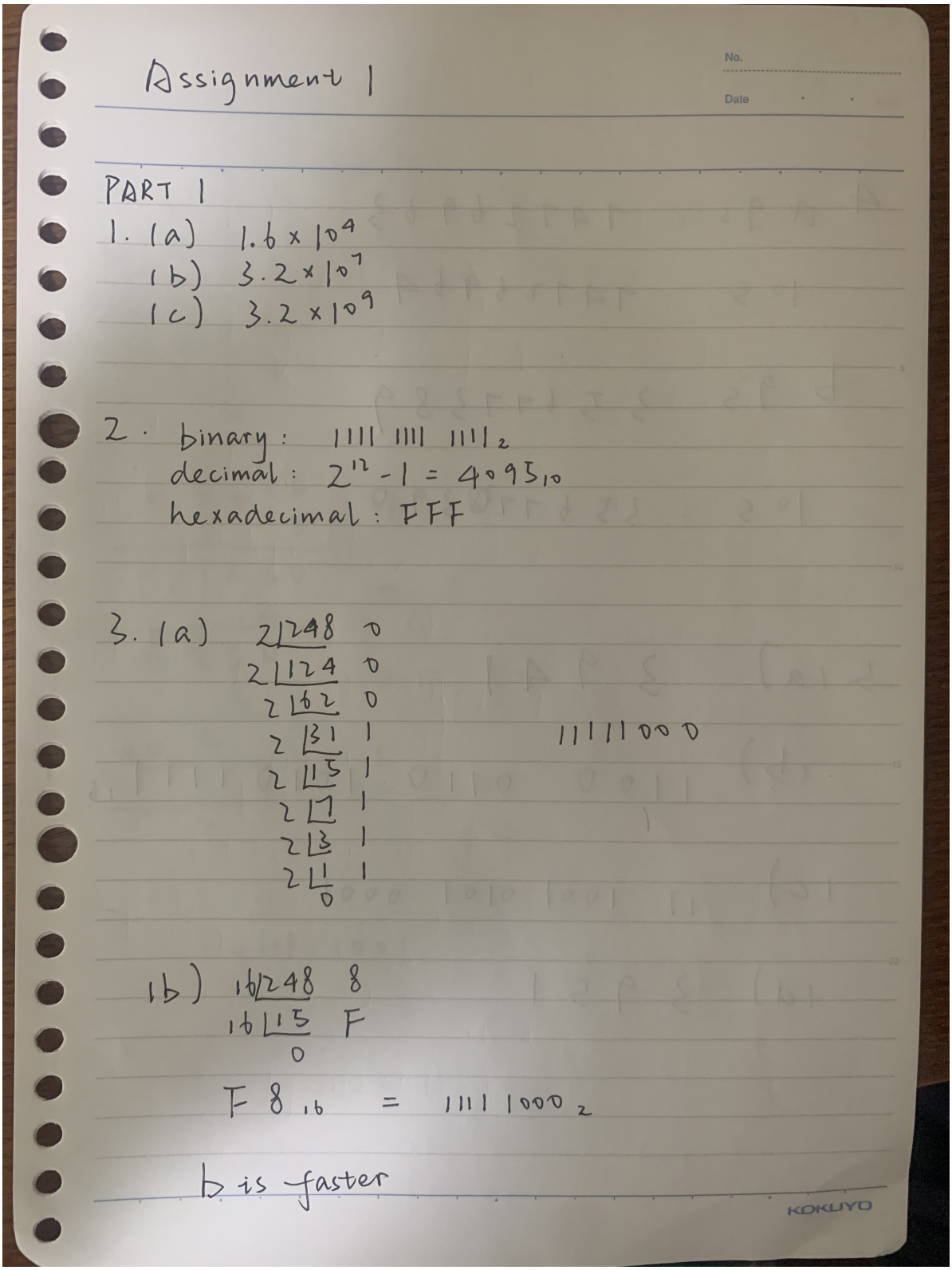
Digital Design - Assignment 1

Name: 张文程 (Zhang Wencheng)

SID: 12010323

Part 1 - DIGITAL DESIGN THEORY



4. a. 9's : 74726963

10's : 74726964

b 9's : 35677389

10's : 35677390

5. 1a) 3941

1b) 1100 0110 1010 1111$_{16}$

1c) 11 1001 0101 0001

1d) 3951

No.

Date

6. 2|23 1

1a) 2|11 1

2|5 1

2|2 0

2|1 1

0

0.5625

× 2

1.1250 1

× 2

0.2500 0

× 2

0.5000 0

× 2

1.0000 1

1011.1001

b. $\frac{5}{3} = 1 + \frac{2}{3}$

2 | 1 ↑
0

$\frac{2}{3} \times 2 = 1 + \frac{1}{3}$

$\frac{1}{3} \times 2 = 0 + \frac{2}{3}$

$\frac{2}{3} \times 2 = 1 + \frac{1}{3}$

⋮

1.1010101

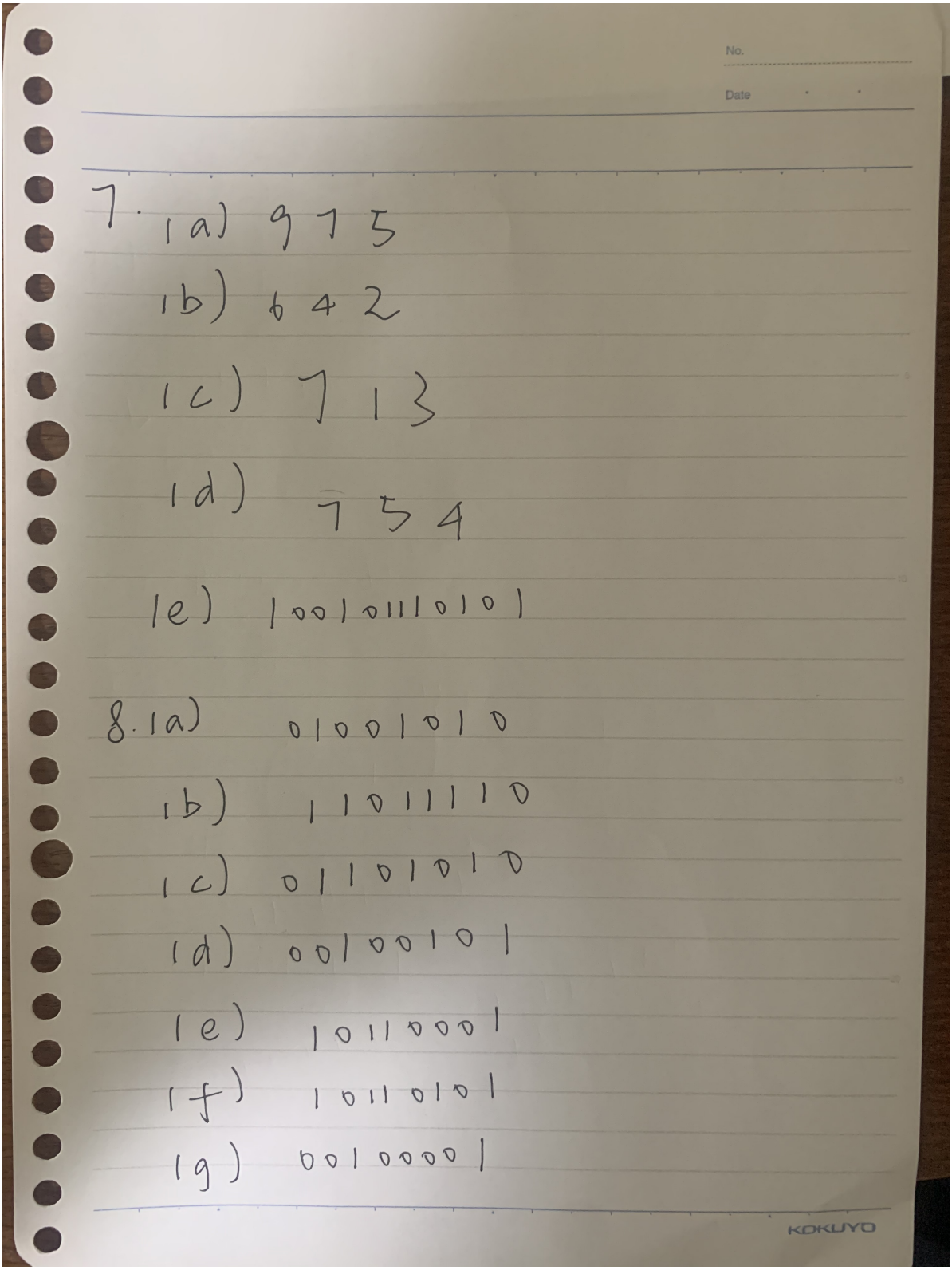
$2^0 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-7} = 1.6640625$

$\frac{5}{3} - 1.6640625 \approx 0.0026$

C. $1.1010101_2 =$ ~~99~~ $1.AA_{16}$

$= 1.6640625_{10}$

the same



Part 2 - DIGITAL DESIGN LAB (Task1)

DESIGN

Code

```
module UnsignedAddition(addend, augend, sum_led);
    parameter WIDTH = 1;
    input [WIDTH-1: 0] addend;
    input [WIDTH-1: 0]  augend;
    output [WIDTH: 0]  sum_led;
    assign sum_led = addend + augend;
endmodule
```

Truth Table

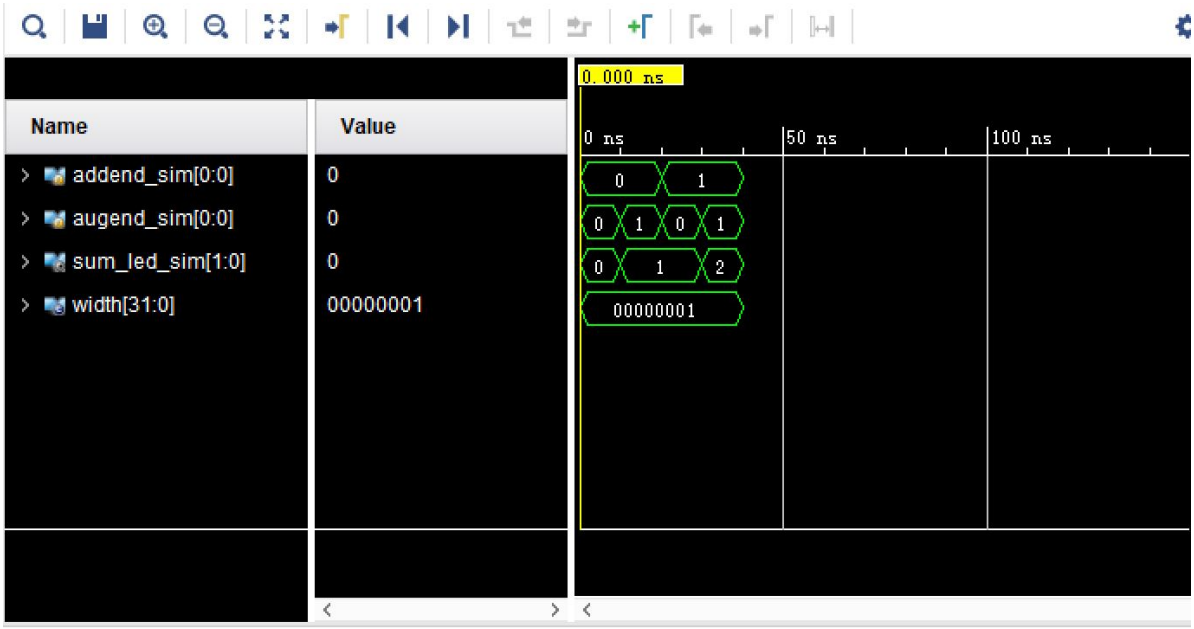
addend	augend	sum_led
0	0	0
0	1	1
1	0	1
1	1	2
00	00	00
00	01	01
00	10	10
00	11	11
01	00	01
01	01	10
01	10	11
01	11	100
10	00	10
10	01	11
10	10	100
10	11	101
11	00	11
11	01	100
11	10	101
11	11	110

SIMULATION

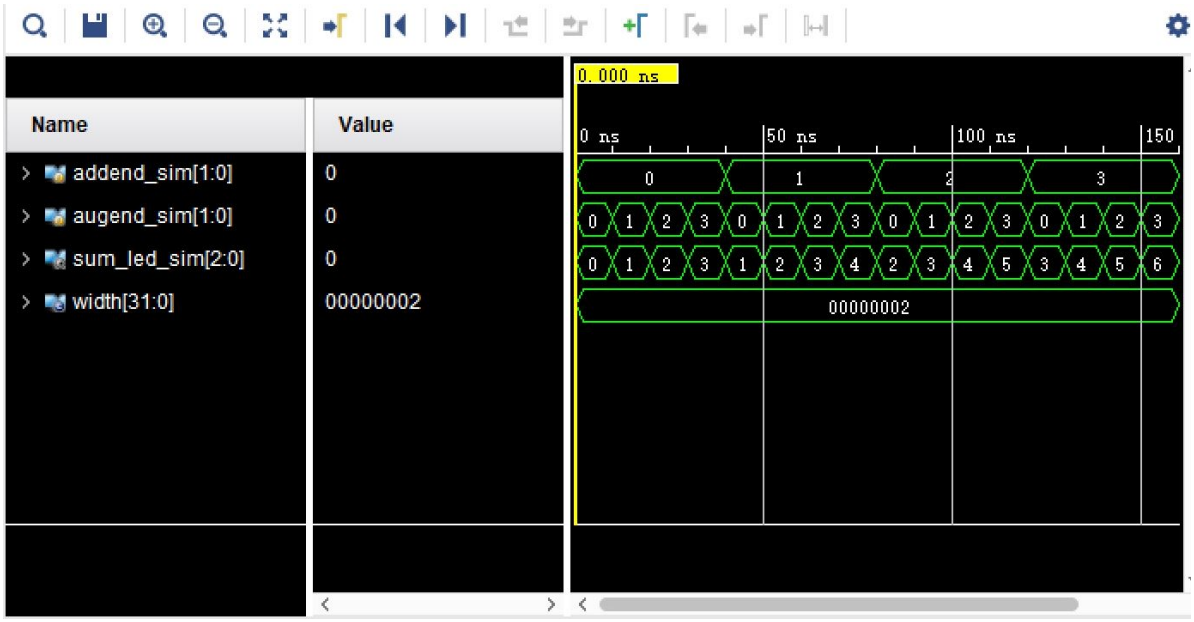
Code

```
module UnsignedAddition_sim();
    parameter width = 1;
    reg [width-1:0] addend_sim;
    reg [width-1:0] augend_sim;
    wire [width:0] sum_led_sim;

    UnsignedAddition #(width) ua(
        .addend(addend_sim),
        .augend(augend_sim),
        .sum_led(sum_led_sim));

    initial begin
        {addend_sim, augend_sim} = 2'b00;
        #10 {addend_sim, augend_sim} = 2'b01;
        #10 {addend_sim, augend_sim} = 2'b10;
        #10 {addend_sim, augend_sim} = 2'b11;
        #10 $finish();
    end

    //initial begin
    //    {addend_sim, augend_sim} = 4'b0000;
    //    #10 {addend_sim, augend_sim} = 4'b0001;
    //    #10 {addend_sim, augend_sim} = 4'b0010;
    //    #10 {addend_sim, augend_sim} = 4'b0011;
    //    #10 {addend_sim, augend_sim} = 4'b0100;
    //    #10 {addend_sim, augend_sim} = 4'b0101;
    //    #10 {addend_sim, augend_sim} = 4'b0110;
    //    #10 {addend_sim, augend_sim} = 4'b0111;
    //    #10 {addend_sim, augend_sim} = 4'b1000;
    //    #10 {addend_sim, augend_sim} = 4'b1001;
    //    #10 {addend_sim, augend_sim} = 4'b1010;
    //    #10 {addend_sim, augend_sim} = 4'b1011;
    //    #10 {addend_sim, augend_sim} = 4'b1100;
    //    #10 {addend_sim, augend_sim} = 4'b1101;
    //    #10 {addend_sim, augend_sim} = 4'b1110;
    //    #10 {addend_sim, augend_sim} = 4'b1111;
    //    #10 $finish();
    //end

endmodule
```

Wave form



Unsigned Addition in 1 bit



Unsigned Addition in 2 bits

Description

The simulation result is the same as the truth-table

Part 3 - DIGITAL DESIGN LAB (Task2)

Design

Data flow code

```
module distributive1bit_df(
    input a,
    input b,
    input c,
    output q1,
    output q2,
    output q3,
    output q4
    );

    assign q1 = a & (b | c);
    assign q2 = (a & b) | (a & c);
    assign q3 = a | (b & c);
    assign q4 = (a | b) * (a | c);
endmodule
```

```
module distributive2bit_df
#(parameter width=2) (
    input [width-1:0] a,
    input [width-1:0] b,
    input [width-1:0] c,
    output [width-1:0] q1,
    output [width-1:0] q2,
    output [width-1:0] q3,
    output [width-1:0] q4
);

    assign q1 = a & (b | c);
    assign q2 = (a & b) | (a & c);
    assign q3 = a | (b & c);
    assign q4 = (a | b) & (a | c);
endmodule
```

Structured design code

```
module distributive1bit_sd(
    input a,
    input b,
    input c,
    output q1,
    output q2,
    output q3,
    output q4
    );
    wire o_borc;
    wire o_aandb;
    wire o_aandc;
    wire o_aorb;
    wire o_aorc;
    wire o_bandc;
```

```
    or or1(o_borc, b, c);
    and and1(q1, a, o_borc);

    and and2(o_aandb, a , b);
    and and3(o_aandc, a, c);
    or or2(q2, o_aandb, o_aandc);

    and and4(o_bandc, b, c);
    or or3(q3, a, o_bandc);

    or or4(o_aorb, a, b);
    or or5(o_aorc, a, c);
    and and5(q4, o_aorb, o_aorc);
endmodule
```

```
module distributive2bit_sd
#(parameter width=2) (
    input [width-1:0] a,
    input [width-1:0] b,
    input [width-1:0] c,
    output [width-1:0] q1,
    output [width-1:0] q2,
    output [width-1:0] q3,
    output [width-1:0] q4
);
    wire [width-1:0] o_borc;
    wire [width-1:0] o_aandb;
    wire [width-1:0] o_aandc;
    wire [width-1:0] o_aorb;
    wire [width-1:0] o_aorc;
    wire [width-1:0] o_bandc;


    or or1(o_borc[0], b[0], c[0]);
    or or1a(o_borc[1], b[1], c[1]);
    and and1(q1[0], a[0], o_borc[0]);
    and and1a(q1[1], a[1], o_borc[1]);

    and and2(o_aandb[0], a[0] , b[0]);
    and and2a(o_aandb[1], a[1] , b[1]);
    and and3(o_aandc[0], a[0], c[0]);
    and and3a(o_aandc[1], a[1], c[1]);
    or or2(q2[0], o_aandb[0], o_aandc[0]);
    or or2a(q2[1], o_aandb[1], o_aandc[1]);

    and and4(o_bandc[0], b[0], c[0]);
    and and4a(o_bandc[1], b[1], c[1]);

    or or3(q3[0], a[0], o_bandc[0]);
    or or3a(q3[1], a[1], o_bandc[1]);

    or or4(o_aorb[0], a[0], b[0]);
    or or4a(o_aorb[1], a[1], b[1]);
    or or5(o_aorc[0], a[0], c[0]);
    or or5a(o_aorc[1], a[1], c[1]);
    and and5(q4[0], o_aorb[0], o_aorc[0]);
```

```
    and and5a(q4[1], o_aorb[1], o_aorc[1]);
endmodule
```

Truth table

a	b	c	q1	q2	q3	q4
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1
00	00	00	00	00	00	00
00	00	01	00	00	00	00
00	00	10	00	00	00	00
00	00	11	00	00	00	00
00	01	00	00	00	00	00
00	01	01	00	00	01	01
00	01	10	00	00	00	00
00	01	11	00	00	01	01
00	10	00	00	00	00	00
00	10	01	00	00	00	00
00	10	10	00	00	10	10
00	10	11	00	00	10	10
00	11	00	00	00	00	00
00	11	01	00	00	01	01
00	11	10	00	00	10	10
00	11	11	00	00	11	11
01	00	00	00	00	00	00
01	00	01	01	01	01	01
01	00	10	00	00	01	01
01	00	11	01	01	01	01
01	01	00	01	01	01	01
01	01	01	01	01	01	01

a	b	c	q1	q2	q3	q4
01	01	10	01	01	01	01
01	01	11	01	01	01	01
01	10	00	00	00	01	01
01	10	01	01	01	01	01
01	10	10	00	00	11	11
01	10	11	01	01	11	11
01	11	00	01	01	01	01
01	11	01	01	01	01	01
01	11	10	01	01	11	11
01	11	11	01	01	11	11
10	00	00	00	00	10	10
10	00	01	00	00	10	10
10	00	10	10	10	10	10
10	00	11	10	10	10	10
10	01	00	00	00	10	10
10	01	01	00	00	11	11
10	01	10	10	10	10	10
10	01	11	10	10	11	11
10	10	00	10	10	10	10
10	10	01	10	10	10	10
10	10	10	10	10	10	10
10	10	11	10	10	10	10
10	11	00	10	10	10	10
10	11	01	10	10	11	11
10	11	10	10	10	10	10
10	11	11	10	10	11	11
11	00	00	00	00	11	11
11	00	01	01	01	11	11
11	00	10	10	10	11	11
11	00	11	11	11	11	11

a	b	c	q1	q2	q3	q4
11	01	00	01	01	11	11
11	01	01	01	01	11	11
11	01	10	11	11	11	11
11	01	11	11	11	11	11
11	10	00	10	10	11	11
11	10	01	11	11	11	11
11	10	10	10	10	11	11
11	10	11	11	11	11	11
11	11	00	11	11	11	11
11	11	01	11	11	11	11
11	11	10	11	11	11	11
11	11	11	11	11	11	11

q1 and q2 are the results of the two sides in the first equation

q3 and q4 are for the second equation

Simulation

Code

```
module main();
    reg a_sim;
    reg b_sim;
    reg c_sim;
    wire q1_sim;
    wire q2_sim;
    wire q3_sim;
    wire q4_sim;

    distributive1bit_df #(1) db1df(
        .a(a_sim),
        .b(b_sim),
        .c(c_sim),
        .q1(q1_sim),
        .q2(q2_sim),
        .q3(q3_sim),
        .q4(q4_sim));

//        distributive1bit_sd db1sd(
//        .a(a_sim),
//        .b(b_sim),
```

```
//        .c(c_sim),
//        .q1(q1_sim),
//        .q2(q2_sim),
//        .q3(q3_sim),
//        .q4(q4_sim));

    initial begin
        {a_sim, b_sim, c_sim} = 3'b000;
        #10 {a_sim, b_sim, c_sim} = 3'b001;
        #10 {a_sim, b_sim, c_sim} = 3'b010;
        #10 {a_sim, b_sim, c_sim} = 3'b011;
        #10 {a_sim, b_sim, c_sim} = 3'b100;
        #10 {a_sim, b_sim, c_sim} = 3'b101;
        #10 {a_sim, b_sim, c_sim} = 3'b110;
        #10 {a_sim, b_sim, c_sim} = 3'b111;
        #10 $finish();
    end

//    reg [1:0] a_sim;
//    reg [1:0] b_sim;
//    reg [1:0] c_sim;
//    wire [1:0] q1_sim;
//    wire [1:0] q2_sim;
//    wire [1:0] q3_sim;
//    wire [1:0] q4_sim;

//    distributive2bit_df  db2df(
//        .a(a_sim),
//        .b(b_sim),
//        .c(c_sim),
//        .q1(q1_sim),
//        .q2(q2_sim),
//        .q3(q3_sim),
//        .q4(q4_sim));

//    distributive2bit_sd #(2) db2sd(
//            .a(a_sim),
//            .b(b_sim),
//            .c(c_sim),
//            .q1(q1_sim),
//            .q2(q2_sim),
//            .q3(q3_sim),
//            .q4(q4_sim));

//    initial begin
//        {a_sim, b_sim, c_sim} = 6'b000000;
//        #10 {a_sim, b_sim, c_sim} = 6'b000001;
//        #10 {a_sim, b_sim, c_sim} = 6'b000010;
//        #10 {a_sim, b_sim, c_sim} = 6'b000011;
//        #10 {a_sim, b_sim, c_sim} = 6'b000100;
//        #10 {a_sim, b_sim, c_sim} = 6'b000101;
//        #10 {a_sim, b_sim, c_sim} = 6'b000110;
//        #10 {a_sim, b_sim, c_sim} = 6'b000111;

//        #10 {a_sim, b_sim, c_sim} = 6'b001000;
//        #10 {a_sim, b_sim, c_sim} = 6'b001001;
//        #10 {a_sim, b_sim, c_sim} = 6'b001010;
//        #10 {a_sim, b_sim, c_sim} = 6'b001011;
```

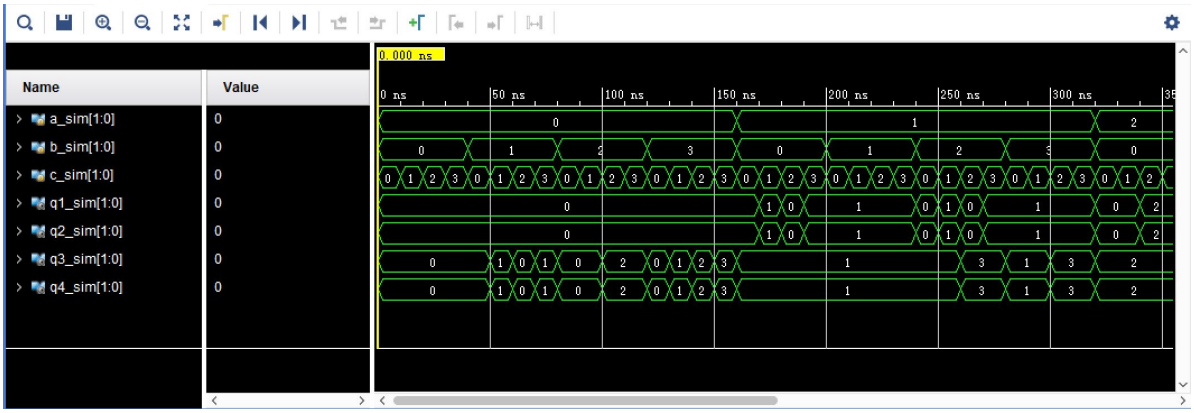
```
//         #10 {a_sim, b_sim, c_sim} = 6'b001100;
//         #10 {a_sim, b_sim, c_sim} = 6'b001101;
//         #10 {a_sim, b_sim, c_sim} = 6'b001110;
//         #10 {a_sim, b_sim, c_sim} = 6'b001111;

//         #10 {a_sim, b_sim, c_sim} = 6'b010000;
//         #10 {a_sim, b_sim, c_sim} = 6'b010001;
//         #10 {a_sim, b_sim, c_sim} = 6'b010010;
//         #10 {a_sim, b_sim, c_sim} = 6'b010011;
//         #10 {a_sim, b_sim, c_sim} = 6'b010100;
//         #10 {a_sim, b_sim, c_sim} = 6'b010101;
//         #10 {a_sim, b_sim, c_sim} = 6'b010110;
//         #10 {a_sim, b_sim, c_sim} = 6'b010111;

//         #10 {a_sim, b_sim, c_sim} = 6'b011000;
//         #10 {a_sim, b_sim, c_sim} = 6'b011001;
//         #10 {a_sim, b_sim, c_sim} = 6'b011010;
//         #10 {a_sim, b_sim, c_sim} = 6'b011011;
//         #10 {a_sim, b_sim, c_sim} = 6'b011100;
//         #10 {a_sim, b_sim, c_sim} = 6'b011101;
//         #10 {a_sim, b_sim, c_sim} = 6'b011110;
//         #10 {a_sim, b_sim, c_sim} = 6'b011111;

//         #10 {a_sim, b_sim, c_sim} = 6'b100000;
//         #10 {a_sim, b_sim, c_sim} = 6'b100001;
//         #10 {a_sim, b_sim, c_sim} = 6'b100010;
//         #10 {a_sim, b_sim, c_sim} = 6'b100011;
//         #10 {a_sim, b_sim, c_sim} = 6'b100100;
//         #10 {a_sim, b_sim, c_sim} = 6'b100101;
//         #10 {a_sim, b_sim, c_sim} = 6'b100110;
//         #10 {a_sim, b_sim, c_sim} = 6'b100111;


//         #10 {a_sim, b_sim, c_sim} = 6'b101000;
//         #10 {a_sim, b_sim, c_sim} = 6'b101001;
//         #10 {a_sim, b_sim, c_sim} = 6'b101010;
//         #10 {a_sim, b_sim, c_sim} = 6'b101011;
//         #10 {a_sim, b_sim, c_sim} = 6'b101100;
//         #10 {a_sim, b_sim, c_sim} = 6'b101101;
//         #10 {a_sim, b_sim, c_sim} = 6'b101110;
//         #10 {a_sim, b_sim, c_sim} = 6'b101111;

//         #10 {a_sim, b_sim, c_sim} = 6'b110000;
//         #10 {a_sim, b_sim, c_sim} = 6'b110001;
//         #10 {a_sim, b_sim, c_sim} = 6'b110010;
//         #10 {a_sim, b_sim, c_sim} = 6'b110011;
//         #10 {a_sim, b_sim, c_sim} = 6'b110100;
//         #10 {a_sim, b_sim, c_sim} = 6'b110101;
//         #10 {a_sim, b_sim, c_sim} = 6'b110110;
//         #10 {a_sim, b_sim, c_sim} = 6'b110111;

//         #10 {a_sim, b_sim, c_sim} = 6'b111000;
//         #10 {a_sim, b_sim, c_sim} = 6'b111001;
//         #10 {a_sim, b_sim, c_sim} = 6'b111010;
//         #10 {a_sim, b_sim, c_sim} = 6'b111011;
//         #10 {a_sim, b_sim, c_sim} = 6'b111100;
//         #10 {a_sim, b_sim, c_sim} = 6'b111101;
//         #10 {a_sim, b_sim, c_sim} = 6'b111110;
```

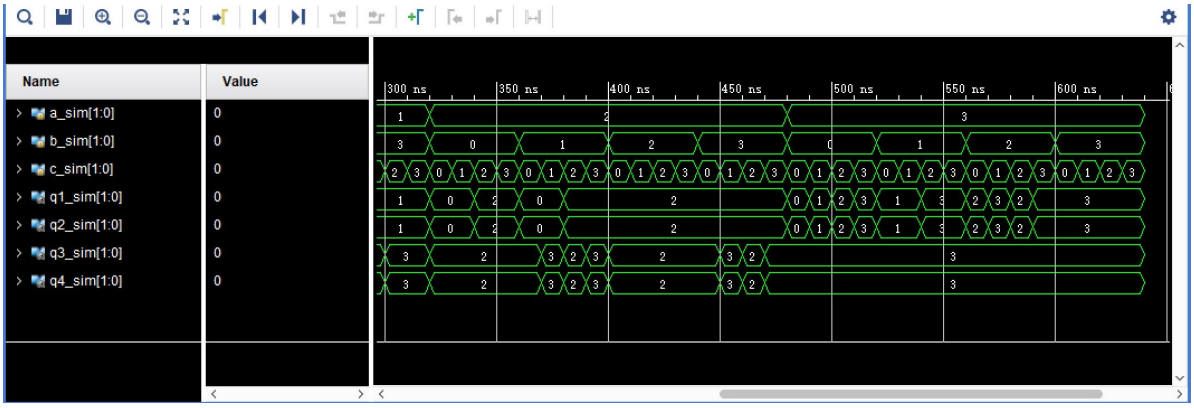
```
//        #10 {a_sim, b_sim, c_sim} = 6'b111111;
//    #10 $finish();
//    end

endmodule
```

Wave form

Distributive1bit_df

Distributive1bit_sd





Distributive2bit_sd

Distributive2bit_sd

Description

The simulation result is the same as the truth-table