



DIGITAL DESIGN

ASSIGNMENT REPORT

ASSIGNMENT ID : 1

Student Name: 侯芳旻

Student ID: 12111448

PART 1: DIGITAL DESIGN THEORY

Provide your answers here:

Box answers when applicable.

Question 1



- (10 points) Convert the decimal number 234.5 to base 3, base 5, base 6, base 12, and base 16. For repeating decimals, write down two reprints (循环节).

(1) $3 \overline{)234}$
 $3 \overline{)78} \ 0$
 $3 \overline{)21} \ 0$
 $3 \overline{)8} \ 2$
 $2 \ 2$
 $0.5 \times 3 = 1.5$
 $0.5 \times 3 = 1.5$
 $(234.5)_{10}$
 $= (22200.11)_3$

(2) $5 \overline{)234}$
 $5 \overline{)76} \ 4$
 $5 \overline{)1} \ 1$
 $1 \ 4$
 $0.5 \times 5 = 2.5$
 $0.5 \times 5 = 2.5$
 $(234.5)_{10}$
 $= (1414.22)_5$

(3) $6 \overline{)234}$
 $6 \overline{)38} \ 0$
 $6 \overline{)6} \ 3$
 $1 \ 0$
 $0.5 \times 6 = 3.0$
 $(234.5)_{10}$
 $= (1030.3)_6$

(4) $12 \overline{)234}$
 $12 \overline{)18} \ 6$
 $1 \ 7$
 $0.5 \times 12 = 6$
 $(234.5)_{10}$
 $= (176.6)_{12}$

(5) $16 \overline{)234}$
 $16 \overline{)14} \ 10$
 $0.5 \times 16 = 8$
 $(234.5)_{10}$
 $= (E A.8)_{16}$

CS207 Assignment 1: Theory (Part 1)

Digital Logic

1 / 8

Question 2



- (5 points) Find the 10's complement of $(791)_{11}$.

$$\begin{aligned} & AAA - 791 + 1 \\ &= 319 + 1 \\ &= (31A)_{11} \end{aligned}$$

CS207 Assignment 1: Theory (Part 1)

Digital Logic

2 / 8

Question 3

- (15 points) Simplify the following Boolean expressions to a minimum number of literals:

- $(a + b + c')(a'b' + c) = ac + bc + a'b'c'$
- $a'b'c + ab'c + abc + a'bc$, and
- $(a + c)(a' + b + c)(a' + b' + c)$.

(1)

| | | | | |
|---|----|----|----|----|
| | bc | | | |
| | 00 | 01 | 11 | 10 |
| a | 0 | 1 | 0 | 0 |
| | 1 | 0 | 1 | 1 |

$$\begin{aligned} & (a + b + c')(a'b' + c) \\ &= ac + bc + a'b'c' \\ &= a'b'c' + bc + ac \end{aligned}$$

(2)

| | | | | |
|---|----|----|----|----|
| | bc | | | |
| | 00 | 01 | 11 | 10 |
| a | 0 | 1 | 1 | 0 |
| | 1 | 0 | 1 | 1 |

$$\begin{aligned} & a'b'c + ab'c + abc + a'bc \\ &= b'c + bc \\ &= c \end{aligned}$$

(3)

| | | | | |
|---|----|----|----|----|
| | bc | | | |
| | 00 | 01 | 11 | 10 |
| a | 0 | 1 | 1 | 0 |
| | 1 | 0 | 1 | 0 |

$$\begin{aligned} & (a + c)(a' + b + c)(a' + b' + c) \\ &= (a + c)(a' + c + b'b) \\ &= (a + c)(a' + c) \\ &= c + aa' \\ &= c \end{aligned}$$

Question 4

- (15 points) Simplify the following three-variable Boolean functions algebraically:

- $F_1(A, B, C) = \sum(1, 2, 6, 7)$,
- $F_2(A, B, C) = \sum(0, 1, 2, 3, 5)$, and
- $F_3(A, B, C) = \sum(3, 5, 6, 7)$.

| | | | | |
|---|----|----|----|----|
| | BC | | | |
| | 00 | 01 | 11 | 10 |
| A | 0 | 1 | 3 | 2 |
| | 1 | 4 | 5 | 6 |

$$\begin{aligned} (1) \quad & F_1(A, B, C) \\ &= A'B'C + A'BC' + ABC' + ABC \\ &= A'B'C + A'BC' + AB \\ &= A'B'C + BC(A + A'C') \\ &= A'B'C + AB + BC' \end{aligned}$$

$$\begin{aligned} (2) \quad & F_2(A, B, C) \\ &= A'B'C' + A'B'C + A'BC' \\ &\quad + A'BC + AB'C \\ &= A'B' + A'B + A'BC \\ &= A' + AB'C \\ &= A' + B'C \end{aligned}$$

$$\begin{aligned} (3) \quad & F_3(A, B, C) \\ &= A'BC + AB'C \\ &\quad + ABC' + A'BC \\ &= A'BC + AB'C \\ &\quad + AB \\ &= A'BC + A(B + B'C) \\ &= A'BC + AB + AC \\ &= BC(A' + A) + AC \\ &= AB + BC + AC \end{aligned}$$

Question 5

- (15 points) Using a Karnaugh map, simplify the following functions:
 - $F_1(A, B, C, D) = \sum(0, 2, 3, 6, 7, 10, 11, 12, 13, 15)$ in sum-of-minterms,
 - $F_2(A, B, C, D) = \sum(1, 9, 10, 12, 13, 14) + d(4, 5, 8)$ in sum-of-minterms, and
 - $F_3(W, X, Y, Z) = \prod(0, 2, 6, 11, 13, 14, 15) + d(1, 3, 9, 10, 12)$ in product-of-maxterms.

(1)

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | 1 | 0 | 1 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 0 | 1 | 1 |

$$F_1(A, B, C, D) = A'B'D + A'C + ABC' + ACD + AB'C$$

CS207 Assignment 1: Theory (Part 1)

(2)

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | 1 | | |
| 01 | X | X | | |
| 11 | 1 | 1 | | 1 |
| 10 | X | 1 | | 1 |

$$F_2(A, B, C, D) = AD' + C'D$$

Digital Logic

(3)

| YZ \ WX | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | 0 | X | X | 0 |
| 01 | | | | 0 |
| 11 | X | 0 | 0 | 0 |
| 10 | | X | 0 | X |

$$F_3(W, X, Y, Z) = (W+X)(Y'+Z)(W'+Z')$$

5 / 8

Question 6

- (10 points) With the use of maps, find the simplest sum-of-products form of the function $F = fg$, where $f = abd' + c'd + a'cd' + b'cd'$ and $g = (a + b + d')(b' + c' + d)(a' + c + d')$.

| cd \ ab | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 1 |

$$F = fg = abc'd' + a'b'cd' + b'cd'$$

CS207 Assignment 1: Theory (Part 1)

Digital Logic

6 / 8

Question 7

- (15 points) Obtain the sum-of-products expression for $F(A, B, C, D) = \sum(1, 2, 4, 7, 8, 9, 11) + d(0, 3, 5)$ and implement it with

- ① NAND gates only, and
- ② NOR gates only.

Draw the two logic diagrams.

| AB \ CD | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | X | 1 | X | 1 |
| 01 | 1 | X | 1 | |
| 11 | | | | |
| 10 | 1 | 1 | 1 | |

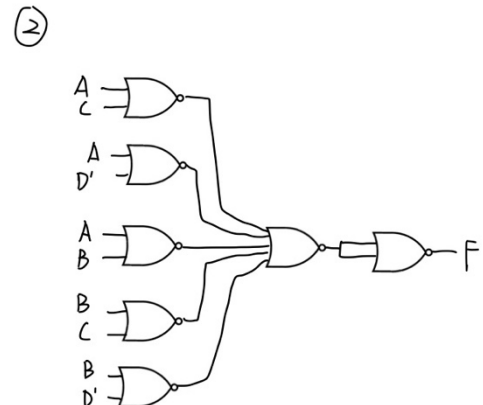
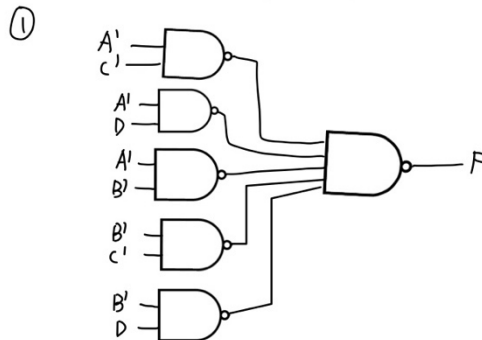
$$F(A, B, C, D) = A'C' + A'D + B'C' + B'D + A'B'$$

Question 7

- (15 points) Obtain the sum-of-products expression for $F(A, B, C, D) = \sum(1, 2, 4, 7, 8, 9, 11) + d(0, 3, 5)$ and implement it with

- ① NAND gates only, and
- ② NOR gates only.

Draw the two logic diagrams.



Question 8

- (15 points) Find the minimized sum-of-products for the logical product $F = F_1 F_2$ of the following pairs of functions:

① $F_1(A, B, C, D) = \sum(1, 3, 5, 7)$

$F_2(A, B, C, D) = \sum(2, 3, 6, 7)$

② $F_1(A, B, C, D) = \sum(1, 3, 5, 6, 8, 10, 11, 12, 13)$

$F_2(A, B, C, D) = \sum(0, 3, 5, 8, 9, 11, 13, 15)$

③ $F_1(A, B, C) = \prod(0, 3, 6, 7)$

$F_2(A, B, C) = \prod(1, 3, 7)$

①

| AB \ CD | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

$F = F_1 \cdot F_2 = A'cD$

②

| AB \ CD | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 10 | 1 | 0 | 1 | 0 |

$F = AB'c'D' + Bc'D + B'cD$

③

| A \ BC | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 |

$F = AB' + A'BC'$

PART 2: DIGITAL DESIGN LAB (TASK1)

DESIGN

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

- Verilog design (provide the Verilog code)

```
module UnsignedMultiplier(  
    input [1:0] in1,  
    input [1:0] in2,  
    output [3:0] out  
);  
    assign out[3] = in1[1]&in1[0]&in2[1]&in2[0];  
    assign out[2] = in1[1]&~in1[0]&in2[1] | in1[1]&in2[1]&~in2[0];  
    assign out[1] = ~in1[1]&in1[0]&in2[1] | in1[0]&in2[1]&~in2[0] | in1[1]&~in2[1]&in2[0] | in1[1]&~in1[0]&in2[0];  
    assign out[0] = in1[0]&in2[0];  
endmodule
```

- *Truth-table*

$$P = in1 * in2$$

| ln1 | ln2 | p |
|-----|-----|------|
| 00 | 00 | 0000 |
| 00 | 01 | 0000 |
| 00 | 10 | 0000 |
| 00 | 11 | 0000 |
| 01 | 00 | 0000 |
| 01 | 01 | 0001 |
| 01 | 10 | 0010 |
| 01 | 11 | 0011 |
| 10 | 00 | 0000 |
| 10 | 01 | 0010 |
| 10 | 10 | 0100 |
| 10 | 11 | 0110 |
| 11 | 00 | 0000 |
| 11 | 01 | 0011 |
| 11 | 10 | 0110 |
| 11 | 11 | 1001 |

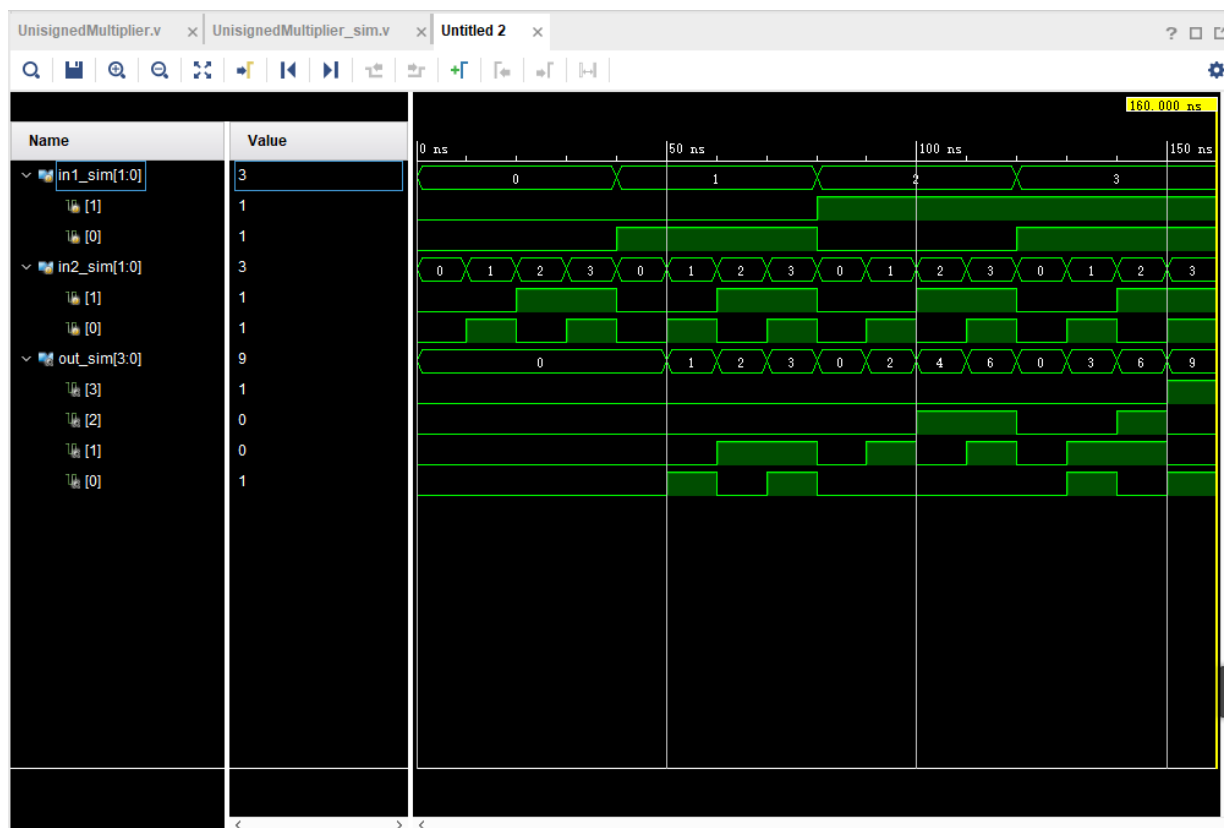
SIMULATION

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

- Using Verilog(provide the Verilog code)

```
22
23 module UnsignedMultiplier_sim();
24   reg [1:0] in1_sim = 2'b00;
25   reg [1:0] in2_sim = 2'b00;
26   wire [3:0] out_sim;
27   UnsignedMultiplier um_sim(in1_sim,in2_sim,out_sim);
28   initial begin
29     repeat(15) #10 {in1_sim,in2_sim} = {in1_sim,in2_sim} + 1;
30     #10 $finish();
31   end
32 endmodule
33
```

- Wave form of simulation result (provide screen shots)



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

| Truth table | | | Simulation result | | | |
|-------------|-----|---------|-------------------|--------|--------|--------|
| in1 | in2 | in1*in2 | out[3] | out[2] | out[1] | out[0] |
| 00 | 00 | 0000 | 0 | 0 | 0 | 0 |
| 00 | 01 | 0000 | 0 | 0 | 0 | 0 |
| 00 | 10 | 0000 | 0 | 0 | 0 | 0 |
| 00 | 11 | 0000 | 0 | 0 | 0 | 0 |
| 01 | 00 | 0000 | 0 | 0 | 0 | 0 |
| 01 | 01 | 0001 | 0 | 0 | 0 | 1 |
| 01 | 10 | 0010 | 0 | 0 | 1 | 0 |
| 01 | 11 | 0011 | 0 | 0 | 1 | 1 |
| 10 | 00 | 0000 | 0 | 0 | 0 | 0 |
| 10 | 01 | 0010 | 0 | 0 | 1 | 0 |
| 10 | 10 | 0100 | 0 | 1 | 0 | 0 |
| 10 | 11 | 0110 | 0 | 1 | 1 | 0 |
| 11 | 00 | 0000 | 0 | 0 | 0 | 0 |
| 11 | 01 | 0011 | 0 | 0 | 1 | 1 |
| 11 | 10 | 0110 | 0 | 1 | 1 | 0 |
| 11 | 11 | 1001 | 1 | 0 | 0 | 1 |

“in1” and “in2” are the 2 input 2-bit unsigned numbers. And “out” is the 4-bit output designed as the production of 2 inputs. Judging from this table, the simulation result is the same as the truth table.

So, the function of the design meet the expectation.

THE DESCRIPTION OF OPERATION

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

- *Problems and solutions*

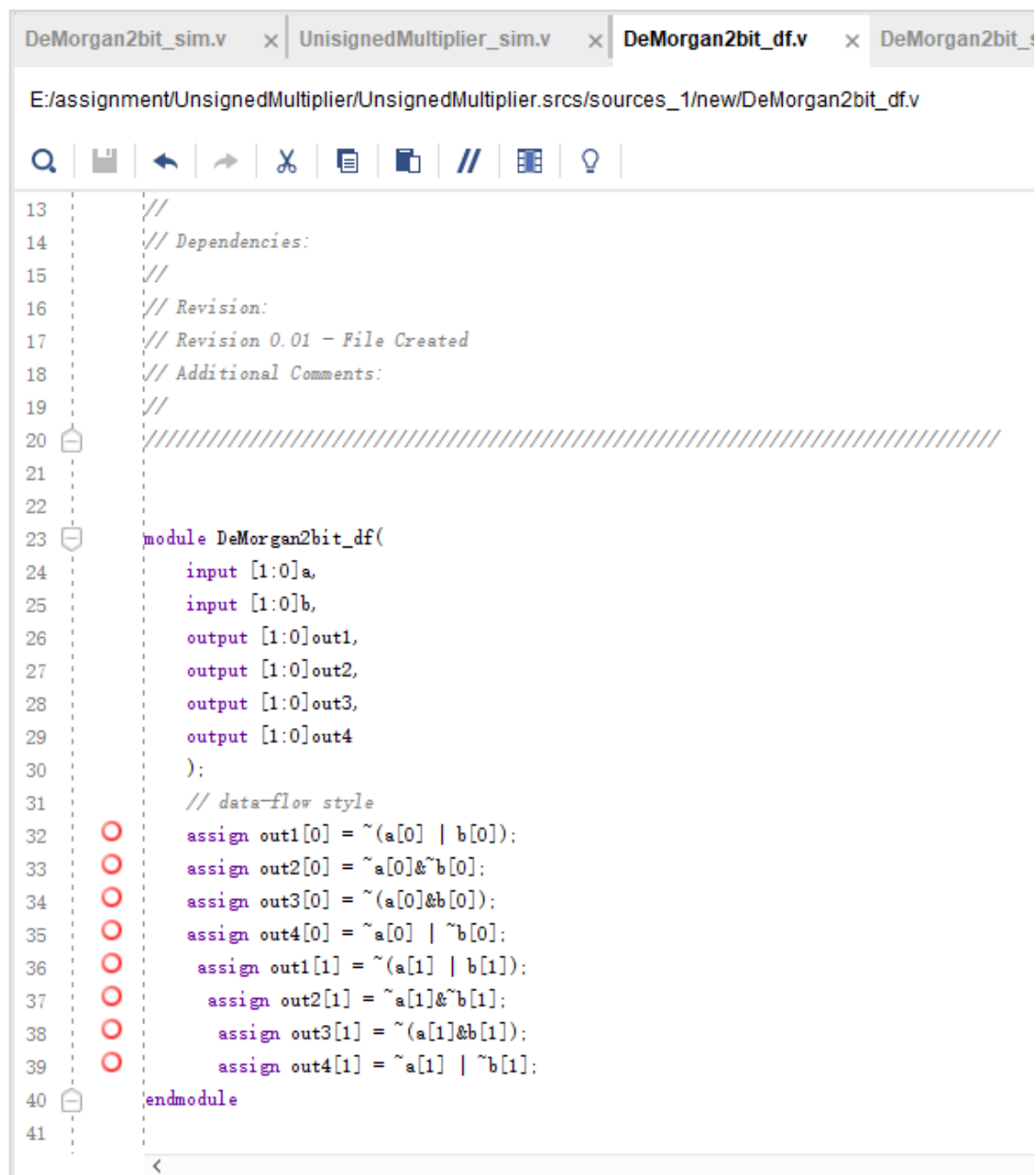
Nothing

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)



```
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module DeMorgan2bit_df(
24     input [1:0]a,
25     input [1:0]b,
26     output [1:0]out1,
27     output [1:0]out2,
28     output [1:0]out3,
29     output [1:0]out4
30 );
31 // data-flow style
32 assign out1[0] = ~(a[0] | b[0]);
33 assign out2[0] = ~a[0]&~b[0];
34 assign out3[0] = ~(a[0]&b[0]);
35 assign out4[0] = ~a[0] | ~b[0];
36 assign out1[1] = ~(a[1] | b[1]);
37 assign out2[1] = ~a[1]&~b[1];
38 assign out3[1] = ~(a[1]&b[1]);
39 assign out4[1] = ~a[1] | ~b[1];
40 endmodule
41
```

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

```

DeMorgan2bit_sd.v
E:/assignment/UnsignedMultiplier/UnsignedMultiplier.srcs/sources_1/new/DeMorgan2bit_sd.v

20 //////////////////////////////////////////////////
21
22
23 module DeMorgan2bit_sd(
24     input [1:0] a,
25     input [1:0] b,
26     output [1:0] out1,
27     output [1:0] out2,
28     output [1:0] out3,
29     output [1:0] out4
30 );
31     // structure-design style
32     wire [1:0] o1out;
33     wire [1:0] a1out;
34     wire [1:0] n2out;
35     wire [1:0] n3out;
36     wire [1:0] a2out;
37     or o10(o1out[0], a[0], b[0]);
38     not n10(out1[0], o1out[0]);
39     not n20(n2out[0], a[0]);
40     // get ~a
41     not n30(n3out[0], b[0]);
42     // get ~b
43     and a10(out2[0], n2out[0], n3out[0]);
44     and a20(a2out[0], a[0], b[0]);
45     not n40(out3[0], a2out[0]);
46     or o20(out4[0], n2out[0], n3out[0]);
47     // module name of primitive gate are lower case
48     // do NOT forget that the primitive gate can only operate on ONE bit
49     or o1(o1out[1], a[1], b[1]);
50     not n1(out1[1], o1out[1]);
51     not n2(n2out[1], a[1]);
52     // get ~a
53     not n3(n3out[1], b[1]);
54     // get ~b
55     and a1(out2[1], n2out[1], n3out[1]);
56     and a2(a2out[1], a[1], b[1]);
57     not n4(out3[1], a2out[1]);
58     or o2(out4[1], n2out[1], n3out[1]);
59 endmodule
60

```

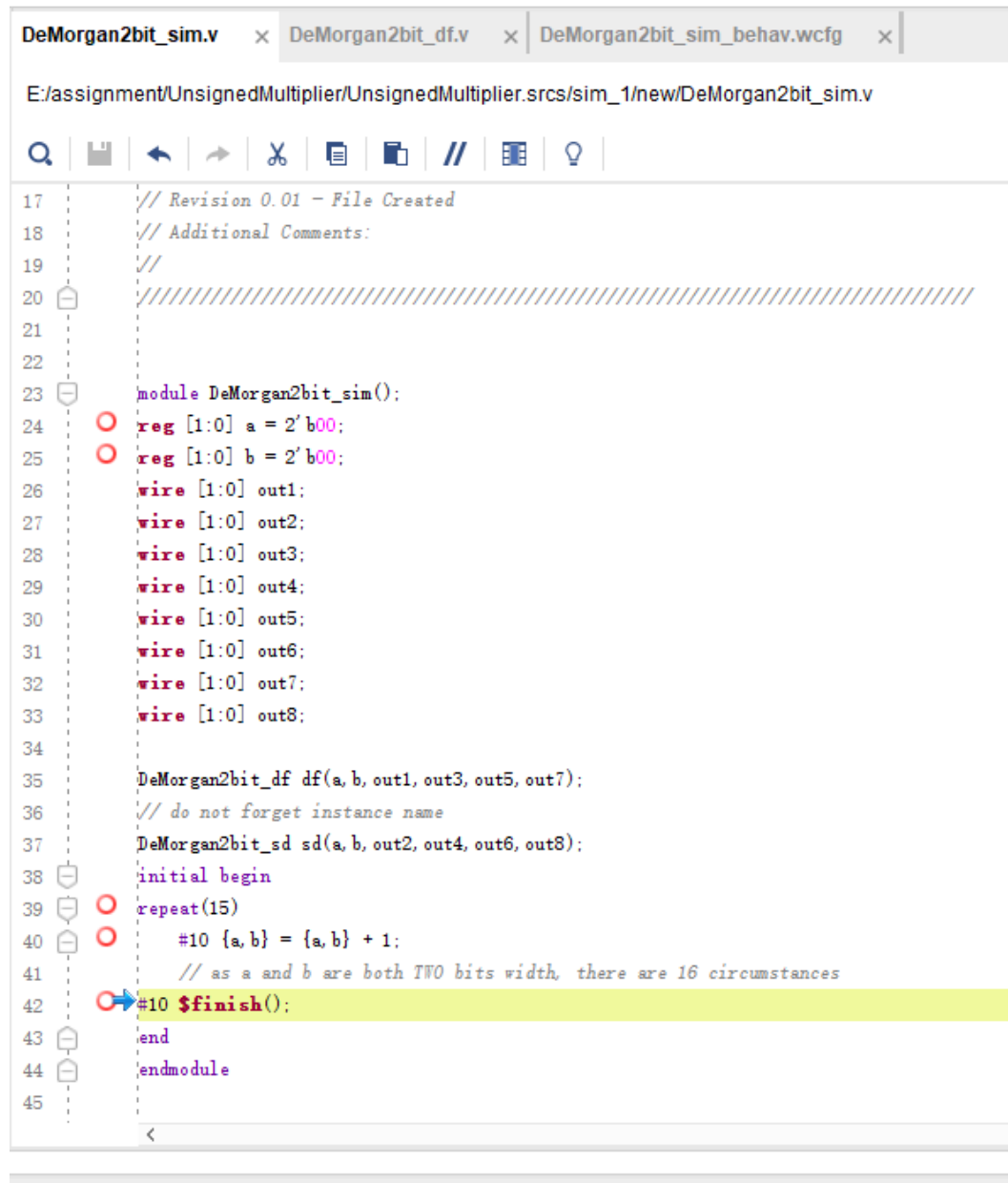
- *Truth-table*

| A | B | $(A+B)'$ | $A' B'$ | $(AB)'$ | $A' + B'$ |
|----|----|----------|---------|---------|-----------|
| 00 | 00 | 11 | 11 | 11 | 11 |
| 00 | 01 | 10 | 10 | 11 | 11 |
| 00 | 10 | 01 | 01 | 11 | 11 |
| 00 | 11 | 00 | 00 | 11 | 11 |
| 01 | 00 | 10 | 10 | 11 | 11 |
| 01 | 01 | 10 | 10 | 10 | 10 |
| 01 | 10 | 00 | 00 | 11 | 11 |
| 01 | 11 | 00 | 00 | 10 | 10 |
| 10 | 00 | 01 | 01 | 11 | 11 |
| 10 | 01 | 00 | 00 | 11 | 11 |
| 10 | 10 | 01 | 01 | 01 | 01 |
| 10 | 11 | 00 | 00 | 01 | 01 |
| 11 | 00 | 00 | 00 | 11 | 11 |
| 11 | 01 | 00 | 00 | 10 | 10 |
| 11 | 10 | 00 | 00 | 01 | 01 |
| 11 | 11 | 00 | 00 | 00 | 00 |

SIMULATION

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

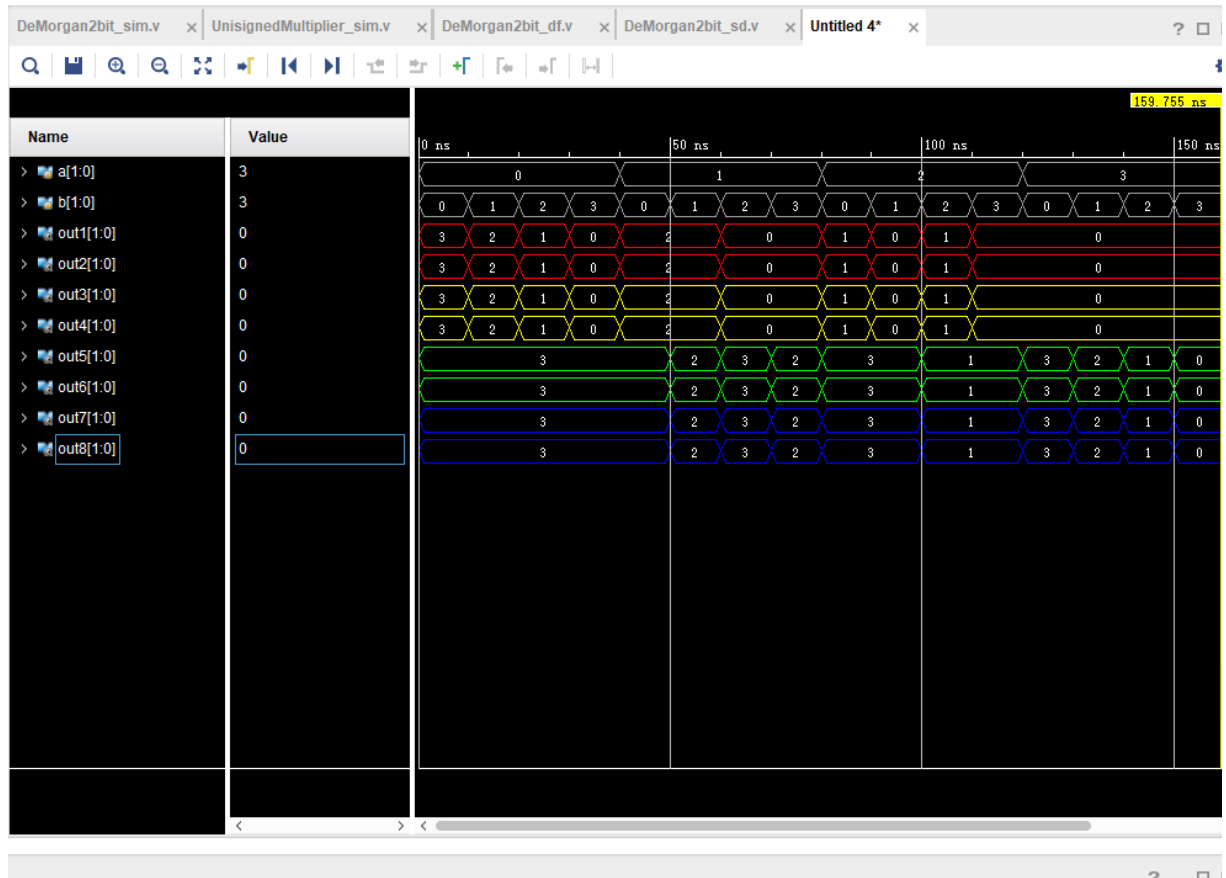
- Using Verilog (provide the Verilog code)



```
DeMorgan2bit_sim.v x DeMorgan2bit_df.v x DeMorgan2bit_sim_behav.wcfg x
E:/assignment/UnsignedMultiplier/UnsignedMultiplier.srcs/sim_1/new/DeMorgan2bit_sim.v

17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module DeMorgan2bit_sim();
24     reg [1:0] a = 2'b00;
25     reg [1:0] b = 2'b00;
26     wire [1:0] out1;
27     wire [1:0] out2;
28     wire [1:0] out3;
29     wire [1:0] out4;
30     wire [1:0] out5;
31     wire [1:0] out6;
32     wire [1:0] out7;
33     wire [1:0] out8;
34
35     DeMorgan2bit_df df(a, b, out1, out3, out5, out7);
36     // do not forget instance name
37     DeMorgan2bit_sd sd(a, b, out2, out4, out6, out8);
38     initial begin
39         repeat(15)
40             #10 {a, b} = {a, b} + 1;
41             // as a and b are both TWO bits width, there are 16 circumstances
42             #10 $finish();
43     end
44 endmodule
45
```

- Wave form of simulation result (provide screen shots)



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

“a” and “b” are the 2-bit-width inputs A and B. So the output “out1”, “out2”, “out3”, “out4”, “out5”, “out6”, “out7”, “out8” are all 2-bit width.

Out1 is the output of $(A+B)'$ using data flow design, out2 is the output of $(A+B)'$ using structured design, out3 is the output of $A'B'$ using data flow design and out4 is the output of $A'B'$ using structured design. These four outputs all equal to the value of $(A+B)'$ and $A'B'$ in the truth table, which suggests that $(A+B)' = A'B'$, and the design using data flow and structured design both meet the expectation.

Out5 is the output of $(AB)'$ using data flow design, out6 is the output of $(AB)'$ using structured design, out7 is the output of $A'+B'$ using data flow design and out8 is the output of $A'+B'$ using structured design. These four outputs all equal

to the value of $(AB)'$ and $A'+B'$ in the truth table, which suggests that $(AB)' = A'+B'$, and the design using data flow and structured design both meet the expectation.

THE DESCRIPTION OF OPERATION

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

- *Problems and solutions*

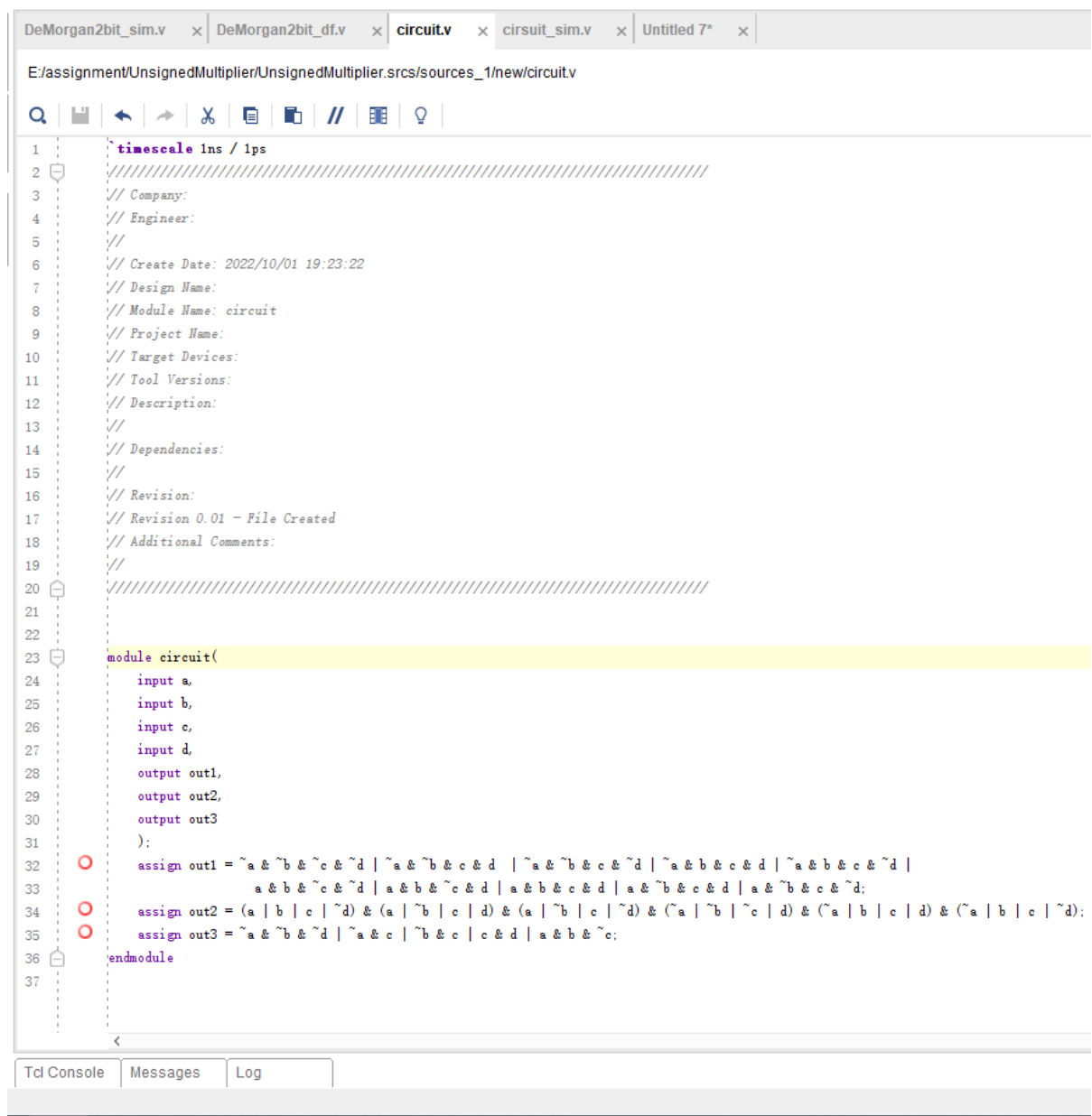
1. Wrongly use the primitive gate as its bitwise is 2 bits. In fact, the bitwise of all primitive gates are 1 bit.
2. Forget to initialize variables of reg type when designing test bench.
3. The primitive gate used to perform negation is called not.
4. While using "initial begin" , do not forget to end it using an "end" .

PART 3: DIGITAL DESIGN LAB (TASK3)

DESIGN

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

- Verilog design (provide the Verilog code) and three expressions



```
1 timescale 1ns / 1ps
2 // Company:
3 // Engineer:
4 //
5 // Create Date: 2022/10/01 19:23:22
6 // Design Name:
7 // Module Name: circuit
8 // Project Name:
9 // Target Devices:
10 // Tool Versions:
11 // Description:
12 //
13 // Dependencies:
14 //
15 // Revision:
16 // Revision 0.01 - File Created
17 // Additional Comments:
18 //
19 //
20 //
21
22
23 module circuit(
24     input a,
25     input b,
26     input c,
27     input d,
28     output out1,
29     output out2,
30     output out3
31 );
32 assign out1 = ~a & ~b & ~c & ~d | ~a & ~b & c & d | ~a & b & c & ~d | ~a & b & c & d |
33             a & b & ~c & ~d | a & b & ~c & d | a & b & c & ~d | a & b & c & d |
34             (a | b | c | ~d) & (a | ~b | c | d) & (a | ~b | c | ~d) & (~a | ~b | ~c | d) & (~a | b | c | d) & (~a | b | c | ~d);
35 assign out2 = ~a & ~b & ~d | ~a & c | ~b & c | c & d | a & b & ~c;
36 endmodule
37
```

Expressions:

Sum of minimum term:

$$F = A' B' C' D' + A' B' CD + A' B' CD' + A' BCD + A' BCD'$$

$$+ ABC' D' + ABC' D + ABCD + AB' CD + AB' CD'$$

Product of maximum term:

$$F = (ABCD') (AB' CD) (AB' CD') (A' B' C' D) (A' BCD) (A' BCD')$$

Simplified:

$$F = A' B' D' + A' C + B' C + CD + ABC'$$

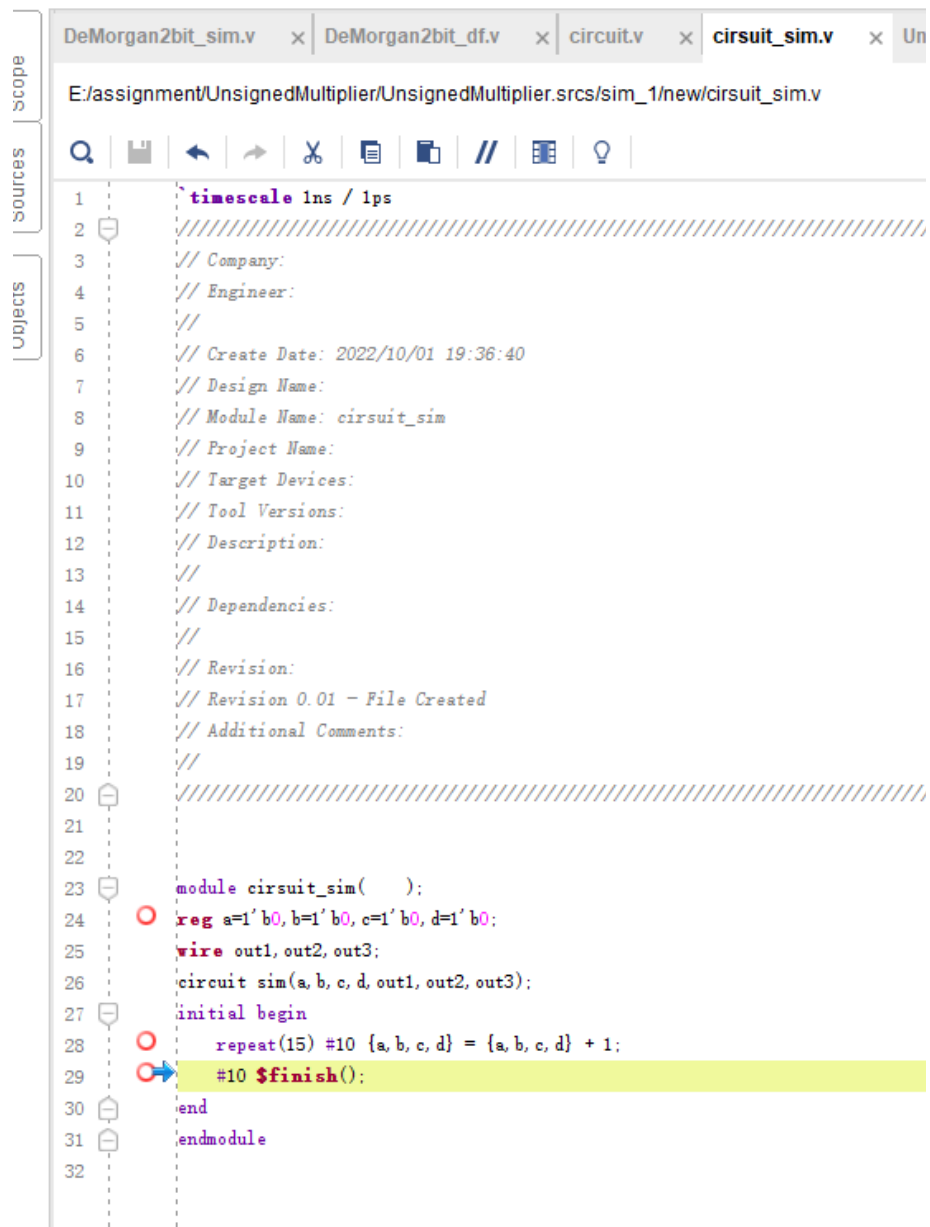
- *Truth-table*

| A | B | C | D | F(A, B, C, D) |
|---|---|---|---|---------------|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

SIMULATION

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

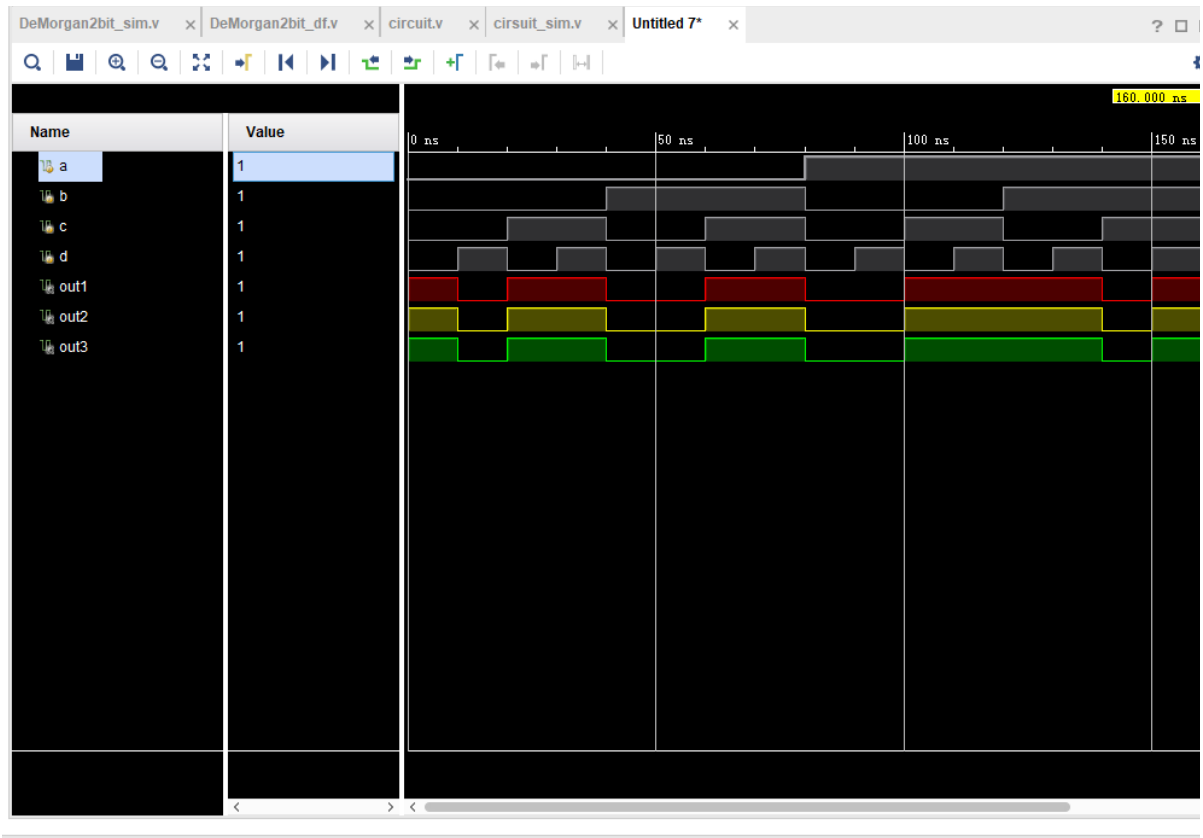
- Using Verilog (provide the Verilog code)



```
DeMorgan2bit_sim.v x DeMorgan2bit_df.v x circuit.v x cirsuit_sim.v x Un
E:/assignment/UnsignedMultiplier/UnsignedMultiplier.srcs/sim_1/new/cirsuit_sim.v

1 timescale 1ns / 1ps
2 //////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 2022/10/01 19:36:40
7 // Design Name:
8 // Module Name: cirsuit_sim
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////
21
22
23 module cirsuit_sim( );
24 reg a=1'b0, b=1'b0, c=1'b0, d=1'b0;
25 wire out1, out2, out3;
26 circuit sim(a, b, c, d, out1, out2, out3);
27 initial begin
28     repeat(15) #10 {a, b, c, d} = {a, b, c, d} + 1;
29     #10 $finish();
30 end
31 endmodule
32
```

- Wave form of simulation result (provide screen shots)



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

“a”, “b”, “c” and “d” are four 1-bit inputs A, B, C and D. “out1”, “out2” and “out3” are 1-bit output.

“out1” is the output of F written as the sum of minimum term, “out2” is the output of F written as the product of maximum item and “out3” is the output of F written as the sum of product obtaining from the Karnaugh Map.

It is clear to see that the simulation results of these 3 outputs are the same, and they are all equal to the value of F in the truth table.

So, the function of the design meet the expectation.

THE DESCRIPTION OF OPERATION

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

- *Problems and solutions*

Module initialization should have an instance name. Don't forget to name it.

Task2-1

| a | b | y1 a' b' | y2 (a+b)' | y3 a'+b' | y4 (ab)' |
|------|------|-------------|--------------|-------------|-------------|
| 00 F | 00 F | T 01 | T 01 | T 01 | T 01 |
| 00 F | 01 T | F 00 | F 00 | T 01 | T 01 |
| 00 F | 10 T | F 00 | F 00 | T 01 | T 01 |
| 00 F | 11 T | F 00 | F 00 | T 01 | T 01 |
| 01 T | 00 F | F 00 | F 00 | T 01 | T 01 |
| 01 T | 01 T | F 00 | F 00 | F 00 | F 00 |
| 01 T | 10 T | F 00 | F 00 | F 00 | F 00 |
| 01 T | 11 T | F 00 | F 00 | F 00 | F 00 |
| 10 T | 00 F | F 00 | F 00 | T 01 | T 01 |
| 10 T | 01 T | F 00 | F 00 | F 00 | F 00 |
| 10 T | 10 T | F 00 | F 00 | F 00 | F 00 |
| 10 T | 11 T | F 00 | F 00 | F 00 | F 00 |
| 11 T | 00 F | F 00 | F 00 | T 01 | T 01 |
| 11 T | 01 T | F 00 | F 00 | F 00 | F 00 |
| 11 T | 10 T | F 00 | F 00 | F 00 | F 00 |
| 11 T | 11 T | F 00 | F 00 | F 00 | F 00 |

Take a,b as boolean type, do the boolean calculation.

- If the value is zero, it is taken as False, if the value is not zero, it is taken as True.
- Using zero to express logic False, one to express logic True.

```

module a1b2_1( input [1:0] a,b, output [1:0]y1,y2,y3,y4 );
assign y1 = !a && !b;
assign y2 = !(a || b);
assign y3 = !a || !b;
assign y4 = !(a && b);
endmodule

```