DESIGN

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

- *Verilog design (provide the Verilog code)*
- *Truth-table*

SIMULATION

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

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

THE DESCRIPTION OF OPERATION

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

Problems and solutions

1.The truth-table while the bit-width of inputs is 2bit.

Task 1 Truth Table

in1[1]	in1[0]	in1[1]	in2[0]	product_led[3]	product_led[2]	product_led[1]	product_led[0]
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

2.Design in Verilog

UnsignedMultiplier.v

```
1  `timescale 1ns / 1ps
2
3  module UnsignedMultiplier(in1, in2, product_led);
4  input [1: 0] in1;
5  input [1: 0] in2;
6  output [3: 0]  product_led;
7      assign product_led[0]=in1[0]&in2[0];
8      assign product_led[1]=(~(in1[1]&in2[0]) & (in1[0]&in2[1]))
9          | (~(in1[0]&in2[1])& (in1[1]&in2[0]));
10     assign product_led[2]= ((in1[1]&in2[0])&~(in1[1]&in2[1])&(in1[0]&in2[1]))|
11         (~(in1[1]&in2[0])&(in1[1]&in2[1])) |((in1[1]&in2[1])&~(in1[0]&in2[1]));
12     assign product_led[3]=(in1[1]&in2[0])&(in1[1]&in2[1]&(in1[0]&in2[1]));
13 endmodule
14
```

vlog?, Line 14, Column 1 Spaces: 4 SystemVerilog

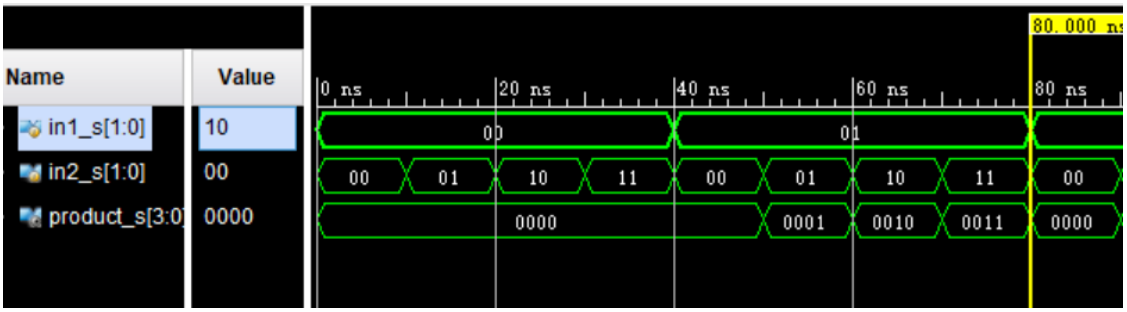
3.Test bench in Verilog

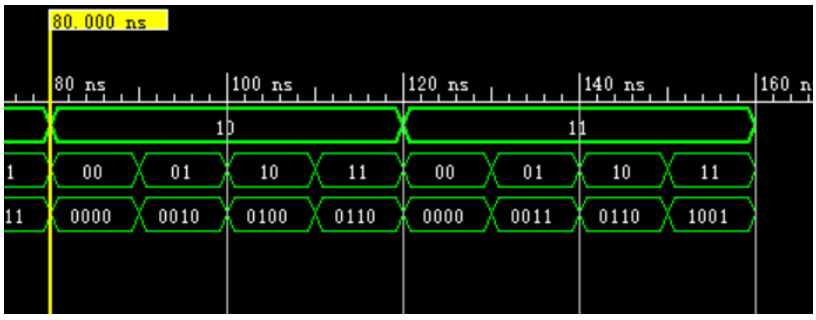
UnsignedMultiplier_sim.v

```
1  `timescale 1ns / 1ps
2
3
4  module UnsignedMultiplier_sim( );
5  reg [1:0]in1_s,in2_s;
6  wire [3:0]product_s;
7  UnsignedMultiplier task1(
8  .in1(in1_s),.in2(in2_s),.product_led(product_s)
9  );
10 initial begin
11     in1_s = 2'b0; in2_s = 2'b0;
12     repeat(15) #10 {in1_s,in2_s} = {in1_s,in2_s} + 1;
13     #10 $finish();
14 end
15 endmodule
16
```

vlog2, Line 16, Column 1 Spaces: 4 SystemVerilog

4.Simulation result





The result is same as the truth table. The function of the design meets the expectation.

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)*
- *Verilog design while using structured design (provide the Verilog code)*
- *Truth-table*

SIMULATION

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

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

THE DESCRIPTION OF OPERATION

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

- *Problems and solutions*

1.The truth-table (2bit bit-width inputs)

In my code, the name of variable:

cos: (A+B)'

poc: A'B'

cop: (AB)'

soc: A'+B'

Task 2

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	01	01
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

2.Design with data-flow style

DeMorgan2bit_df.v | DeMorgan2bit_sd.v | DeMorgan2bit_sim.v

```
1  `timescale 1ns / 1ps
2
3  module DeMorgan2bit_df(a, b, cos, poc, cop, soc);
4  input [1: 0] a,b;
5  output [1: 0] cos, poc, cop, soc;
6      assign cos[0]=~(a[0]|b[0]);
7      assign cos[1]=~(a[1]|b[1]);
8      assign poc[0]=~a[0]&~b[0];
9      assign poc[1]=~a[1]&~b[1];
10     assign cop[0]=~(a[0]&b[0]);
11     assign cop[1]=~(a[1]&b[1]);
12     assign soc[0]=~a[0]|~b[0];
13     assign soc[1]=~a[1]|~b[1];
14 endmodule
15
```

vlog?, Line 15, Column 1 Spaces: 4 SystemVerilog

3.Design with structure-design style

DeMorgan2bit_df.v | DeMorgan2bit_sd.v | DeMorgan2bit_sim.v

```
1  `timescale 1ns / 1ps
2
3  module DeMorgan2bit_sd(a, b, cos, poc, cop, soc);
4  input [1: 0] a,b;
5  output [1: 0] cos, poc, cop, soc;
6  wire ou1,ou2,ou3,ou4;
7      not (ou1,a[0]);
8      not (ou2,b[0]);
9      not (ou3,a[1]);
10     not (ou4,b[1]);
11     nor (cos[0],a[0],b[0]);
12     nor (cos[1],a[1],b[1]);
13     and (poc[0],ou1,ou2);
14     and (poc[1],ou3,ou4);
15     nand (cop[0],a[0],b[0]);
16     nand (cop[1],a[1],b[1]);
17     or (soc[0],ou1,ou2);
18     or (soc[1],ou3,ou4);
19 endmodule
20
```

vlog?, Line 20, Column 1 Spaces: 4 SystemVerilog

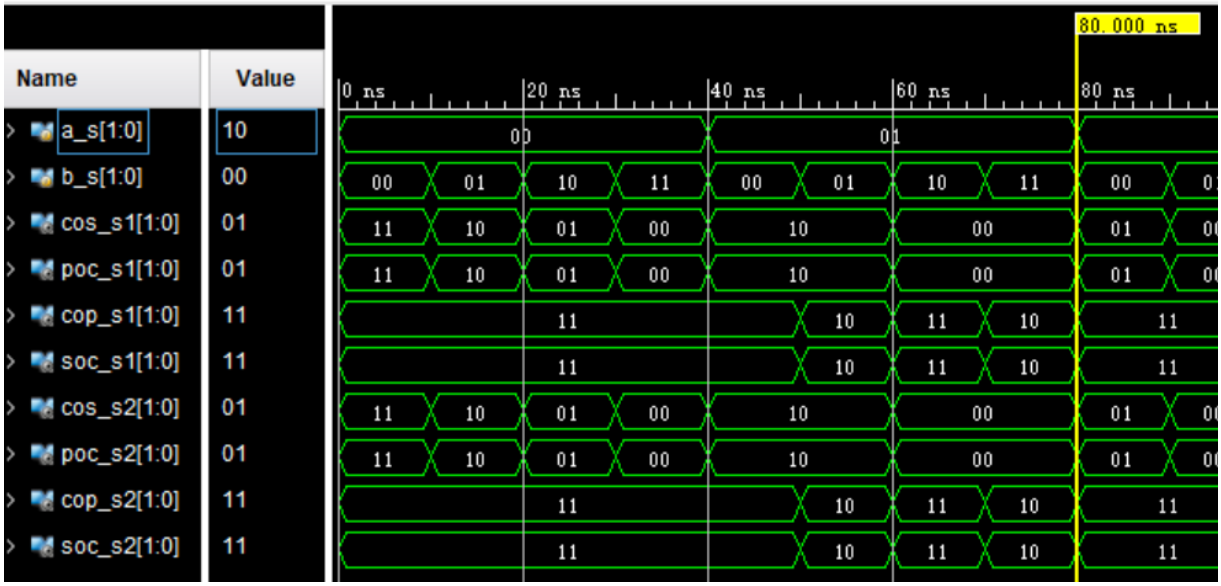
4.Test bench in Verilog (1 testbench is enough)

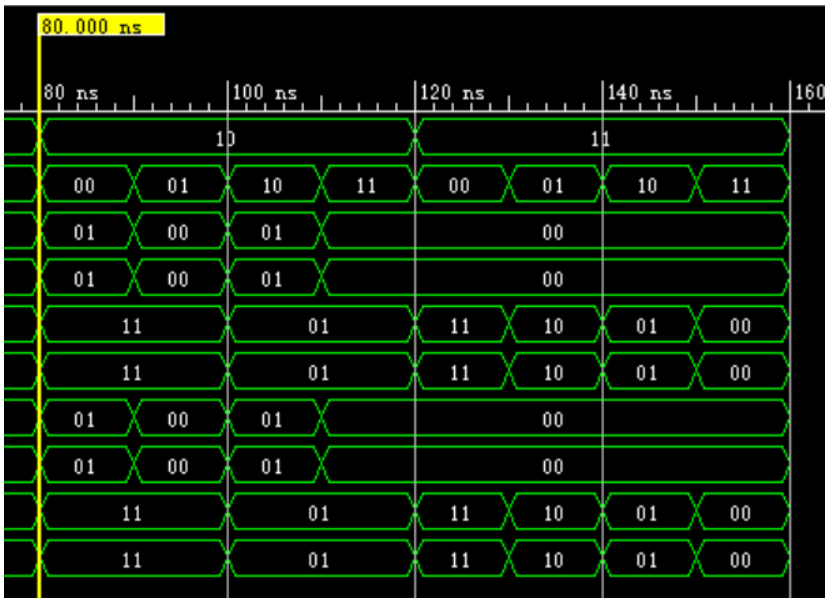
DeMorgan2bit_df.v | DeMorgan2bit_sd.v | DeMorgan2bit_sim.v

```
1  `timescale 1ns / 1ps
2
3  module DeMorgan2bit_sim( );
4  reg [1: 0] a_s,b_s;
5  wire [1: 0] cos_s1, poc_s1, cop_s1, soc_s1, cos_s2, poc_s2, cop_s2, soc_s2;
6  DeMorgan2bit_df df(
7  .a(a_s),.b(b_s),.cos(cos_s1),.poc(poc_s1),.cop(cop_s1),.soc(soc_s1)
8  );
9  DeMorgan2bit_sd sd(
10 .a(a_s),.b(b_s),.cos(cos_s2),.poc(poc_s2),.cop(cop_s2),.soc(soc_s2)
11 );
12 initial begin
13     a_s = 2'b0; b_s = 2'b0;
14     repeat(15) #10 {a_s,b_s} = {a_s,b_s} + 1;
15     #10 $finish( );
16 end
17 endmodule
18
```

vlog?, Line 18, Column 1 Spaces: 4 SystemVerilog

5.simulation result, and answer about the DeMorgan theorem





Same as the truth table,

The function of the design meets the expectation

and De Morgan's law in 2 bits is right. As the (A+B)'=A'B', (AB)'=A'+B'.

PART 2: DIGITAL DESIGN LAB (TASK3)

1.The 3 expressions and Design with data-flow style

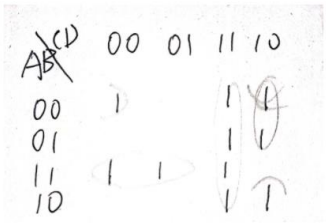
$$F(A,B,C,D) = \sum(0,2,3,6,7,10,11,12,13,15)$$

$$= A'B'C'D' + A'B'CD' + A'B'CD + A'BCD' + A'BCD + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD$$

$$F(A,B,C,D) = \prod(1,4,5,8,9,14)$$

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

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



Task3.v | Task3_sim.v

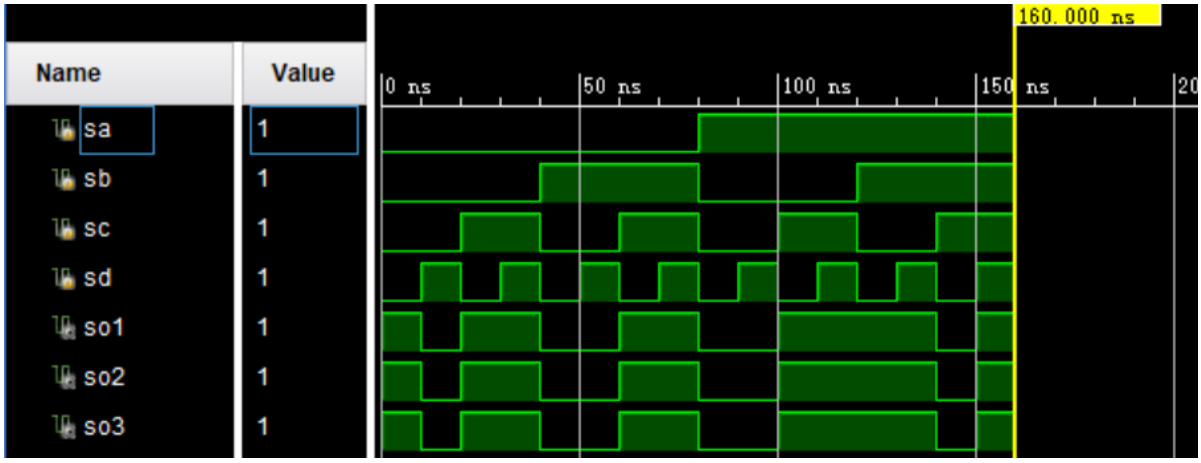
```
1  `timescale 1ns / 1ps
2
3  module Task3(a, b, c, d, o1, o2, o3);
4  input  a, b, c, d;
5  output  o1, o2, o3;
6      assign o1 = (~a&~b&~c&~d)|(~a&~b&c&~d)|(~a&~b&c&d)|(~a&b&c&~d)|(~a&b&c&d)
7          |(a&~b&c&~d)|(a&~b&c&d)|(a&b&~c&~d)|(a&b&~c&d)|(a&b&c&d);
8      assign o2 = (a|b|c|~d)&(a|~b|c|d)&(a|~b|c|~d)&(~a|b|c|d)&(~a|b|c|~d)&(~a|~b|~c|d);
9      assign o3 = (~b&c&~d)|(~a&c&~d)|(~a&~b&~d)|(a&b&~c)|(c&d);
10 endmodule
11
```

2.Test bench in Verilog (1 testbench is enough)

Task3.v | Task3_sim.v

```
1  `timescale 1ns / 1ps
2
3
4  module Task3_sim( );
5  reg sa, sb, sc, sd;
6  wire so1, so2, so3;
7  Task3 task3(
8      sa, sb, sc, sd, so1, so2, so3
9  );
10 initial begin
11     sa = 0; sb = 0; sc = 0; sd = 0;
12     repeat(15) #10 {sa, sb, sc, sd} = {sa, sb, sc, sd} + 1;
13     #10 $finish( );
14 end
15 endmodule
16
```

3.Simulation result(waveform and its description)



The function of the design meets the expectation.

They are actually same for out1, out2, out3.