DIGITAL DESIGN

ASSIGNMENTREPORT

ASSIGNMENT ID : 3

Student Name: 徐驰 (Xu Chi)

Student ID: 11912224

PART 1: DIGITAL DESIGN THEORY

Provide your answers here：

1. F(A, B, C, D) = ∑ (1, 3, 5, 8, 9, 10, 11, 13, 15)

K-map

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

AND-OR: F = AB’ + C’D + AD + B’D

OR-AND: F = (A + D) (B’ + D) (A + B’ + C’)

NOR-OR: F = (A’ + B)’ + (C + D’)’ + (A’ + D’)’ + (B + D’)’

NOR-NOR: F = ((A + D)’ + (B’ + D)’ + (A + B’ + C’)’)’

NAND-NAND: F = ((AB’)’ (C’D)’ (AD)’ (B’D)’)’

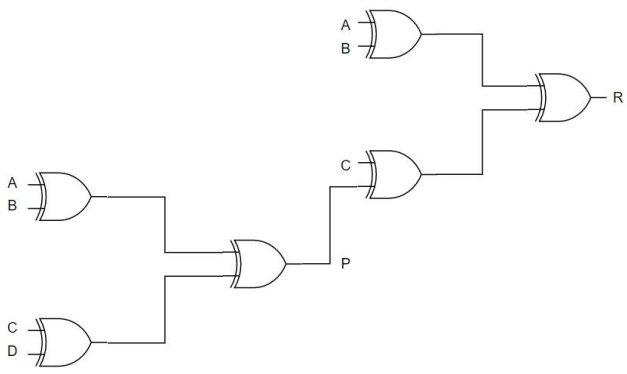
NAND-AND: F = (A’D’)’ (BD’)’ (A’BC)’

2.

parity generator					parity checker				
input				output1	input				output
A	B	C	D	P	A	B	C	P	R

0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	1
0	0	1	0	1	0	0	1	1	0
0	0	1	1	0	0	0	1	0	1
0	1	0	0	1	0	1	0	1	0
0	1	0	1	0	0	1	0	0	1
0	1	1	0	0	0	1	1	0	0
0	1	1	1	1	0	1	1	1	1
1	0	0	0	1	1	0	0	1	0
1	0	0	1	0	1	0	0	0	1
1	0	1	0	0	1	0	1	0	0
1	0	1	1	1	1	0	1	1	1
1	1	0	0	0	1	1	0	0	0
1	1	0	1	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	0
1	1	1	1	0	1	1	1	0	1

F = (A ⊕ B ⊕ C ⊕ D) ⊕ A ⊕ B ⊕ C = D



3.

input			output		
x	y	z	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

F = {A, B, C}

A:x\yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1

A = xy + xz + yz

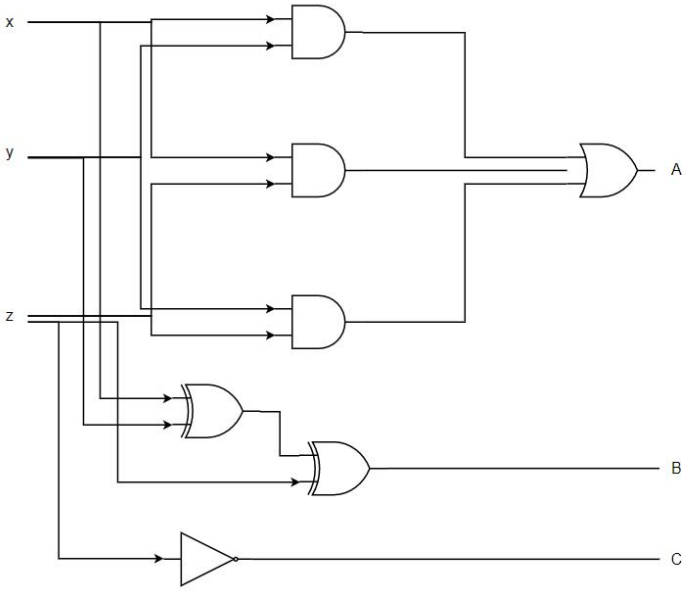
B:x\yz	00	01	11	10
0	0	1	0	1
1	1	0	1	0

B = x ⊕ y ⊕ z

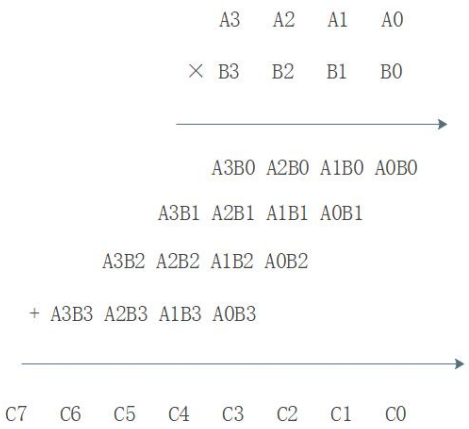
C:x\yz	00	01	11	10
0	1	0	0	1

1	1	0	0	1

C = z’



4. Here is the principle:



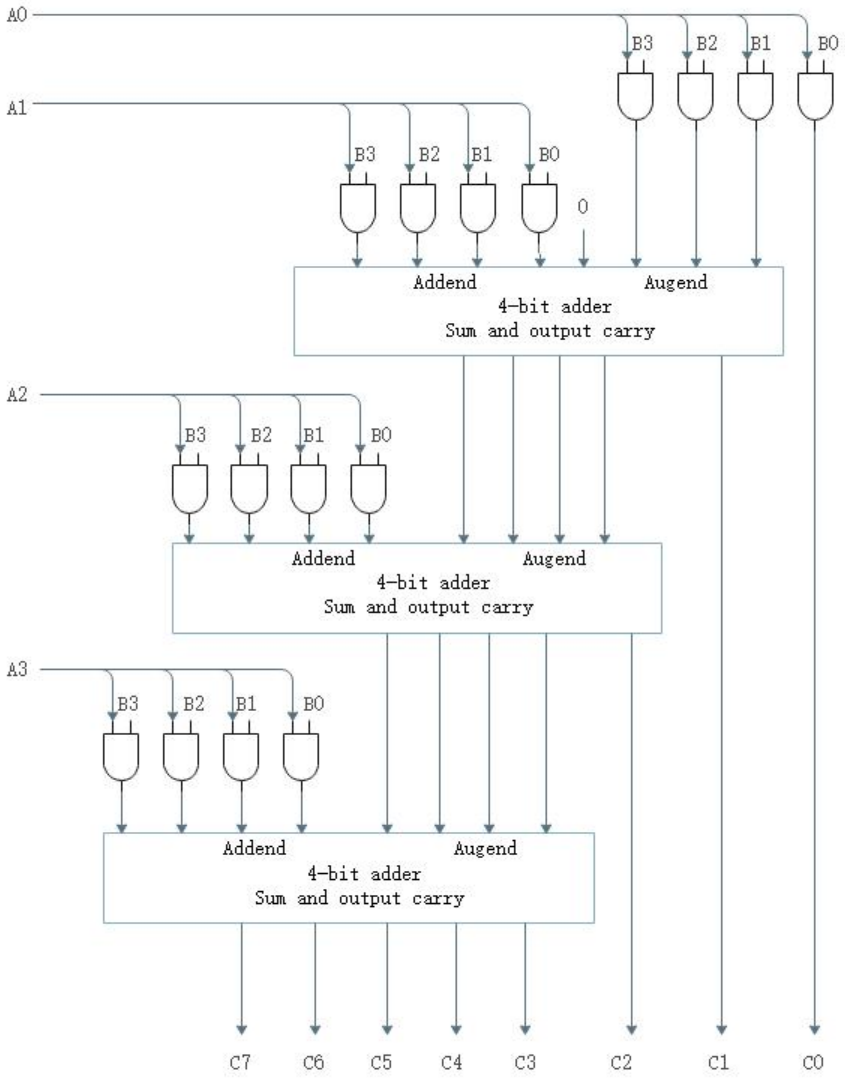
C = A3A2A1A0 × B3B2B1B0

= {C7, C6, C5, C4, C3, C2, C1, C0}

C0 = A0B0, C1 = A1B0 + A0B1, C2 = A2B0 + A1B1 + A0B2, C3 = A3B0 + A2B1 + A1B2 + A0B3,

C4 = A3B1 + A2B2 + A1B3, C5 = A3B2 + A2B3, C6 = A3B3 NB: Ci is not concluded carry bit!

Here is the circuit diagram:



5.

```
module mux
(EN,S3,S2,S1,S0,D15,D14,D13,D12,D11,D10,D9,D8,D7,D6,D5,D4,D3,D2,D1,D0,Y,W);
```

```
input EN,S3,S2,S1,S0,D15,D14,D13,D12,D11,D10,D9,D8,D7,D6,D5,D4,D3,D2,D1,D0;

output reg Y;

output W;

always @*

if (~EN)

      case({S3,S2,S1,S0})

            4'b0000: Y = D0;

            4'b0001: Y = D1;

            4'b0010: Y = D2;

            4'b0011: Y = D3;

            4'b0100: Y = D4;

            4'b0101: Y = D5;

            4'b0110: Y = D6;

            4'b0111: Y = D7;

            4'b1000: Y = D8;

            4'b1001: Y = D9;

            4'b1010: Y = D10;

            4'b1011: Y = D11;

            4'b1100: Y = D12;

            4'b1101: Y = D13;
```

```
        4'b1110: Y = D14;
        4'b1111: Y = D15;
    endcase
else
    Y = 1'b0;
assign W = ~Y;
endmodule
```

(a) F(A, B, C, D) =∑ (1, 2, 5, 8, 10, 14)

= 0 * A'B'C'D'+ 1 * A'B'C'D + 1 * A'B'CD' + 0 * A'B'CD + 0 * A'BC'D' + 1 * A'BC'D + 0 * A'BCD' + 0 * A'BCD + 1 * AB'C'D' + 0 * AB'C'D + 1 * AB'CD' + 0 * AB'CD + 0 * ABC'D' + 0 * ABC'D + 1 * ABCD' + 0 * ABCD

= D0 * m0 + D1 * m1 + D2 * m2 + D3 * m3 + D4 * m4 + D5 * m5 + D6 * m6 + D7 * m7 + D8 * m8 + D9 * m9 + D10 * m10 + D11 * m11 + D12 * m12 + D13 * m13 + D14 * m14 + D15 * m15

= m0 * 0 + m1 * 1 + m2 * 1 + m3 * 0 + m4 * 0 + m5 * 1 + m6 * 0 + m7 * 0 + m8 * 1+ m9 * 0 + m10 * 1 + m11 * 0 + m12 * 0 + m13 * 0 + m14 * 1 + m15 * 0

```
module fun_abcd_mux(input A,B,C,D,output F);
wire sen,sd15,sd14,sd13,sd12,sd11,sd10,sd9,sd8,sd7,sd6,sd5,sd4,sd3,sd2,sd1,sd0;
wire snf;
assign {sen,sd15,sd14,sd13,sd12,sd11,sd10,sd9,sd8,sd7,sd6,sd5,sd4,sd3,sd2,sd1,sd0} =
17'b0_0100_0101_0010_0110;
```

```
mux xc(.EN(sen),

.S3(A),.S2(B),.S1(C),.S0(D),

.D15(sd15),.D14(sd14),.D13(sd13),.D12(sd12),

.D11(sd11),.D10(sd10),.D9(sd9),.D8(sd8),

.D7(sd7),.D6(sd6),.D5(sd5),.D4(sd4),

.D3(sd3),.D2(sd2),.D1(sd1),.D0(sd0),

.Y(F),.W(snf));

endmodule
```

(b) F(A, B, C, D) = Π (4, 5, 11) = ∑ (0, 1, 2, 3, 6, 7, 8, 9, 10, 12, 13, 14, 15)

= 1 * A'B'C'D'+ 1 * A'B'C'D + 1 * A'B'CD' + 1 * A'B'CD + 0 * A'BC'D' + 0 * A'BC'D + 1 * A'BCD' + 1 * A'BCD + 1 * AB'C'D' + 1 * AB'C'D + 1 * AB'CD' + 0 * AB'CD + 1 * ABC'D' + 1 * ABC'D + 1 * ABCD' + 1 * ABCD

= D0 * m0 + D1 * m1 + D2 * m2 + D3 * m3 + D4 * m4 + D5 * m5 + D6 * m6 + D7 * m7 + D8 * m8 + D9 * m9 + D10 * m10 + D11 * m11 + D12 * m12 + D13 * m13 + D14 * m14 + D15 * m15

= m0 * 1 + m1 * 1 + m2 * 1 + m3 * 1 + m4 * 0 + m5 * 0 + m6 * 1 + m7 * 1 + m8 * 1+ m9 * 1 + m10 * 1 + m11 * 0 + m12 * 1 + m13 * 1 + m14 * 1 + m15 * 1

```
module fun_abcd_mux(input A,B,C,D,output F);

wire sen,sd15,sd14,sd13,sd12,sd11,sd10,sd9,sd8,sd7,sd6,sd5,sd4,sd3,sd2,sd1,sd0;

wire snf;

assign {sen,sd15,sd14,sd13,sd12,sd11,sd10,sd9,sd8,sd7,sd6,sd5,sd4,sd3,sd2,sd1,sd0} =
```

```
17'b0_1111_0111_1100_1111;

mux xc(.EN(sen),

.S3(A),.S2(B),.S1(C),.S0(D),

.D15(sd15),.D14(sd14),.D13(sd13),.D12(sd12),

.D11(sd11),.D10(sd10),.D9(sd9),.D8(sd8),

.D7(sd7),.D6(sd6),.D5(sd5),.D4(sd4),

.D3(sd3),.D2(sd2),.D1(sd1),.D0(sd0),

.Y(F),.W(snf));

endmodule
```

6. Since there are two outputs(Dif and Bo), we have to select 2 multiplexers. Start with the truth table of full subtractor. Use karnaugh maps. Select 2 variables as your select line. For example x and y in my case.

The truth table:

input			output	
x	y	z	Dif	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0

1	1	0	0	0
1	1	1	1	1

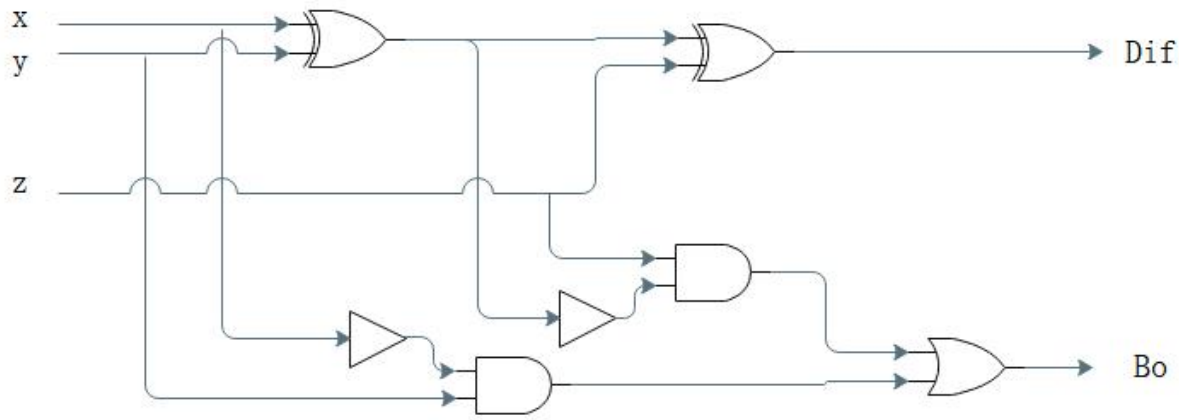
With K-map:

Dif:x\yz	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Dif = x ⊕ y ⊕ z

Bo:x\yz	00	01	11	10
0	0	1	1	1
1	0	0	1	0

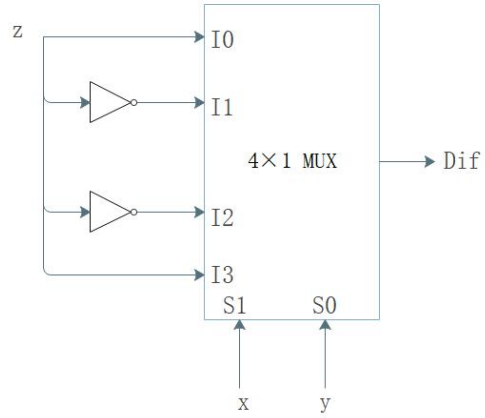
Bo = x'z + x'y + yz

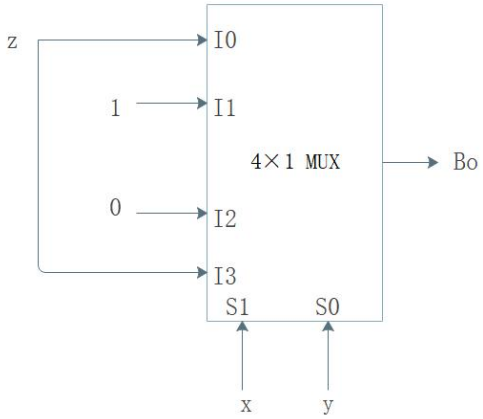


Implement by two 4×1 multiplexers

input			output	
x	y	z	Dif	Bo
0	0	0	0=z	0=z

0	0	1	1=z	1=z
0	1	0	1=z’	1=1
0	1	1	0=z’	1=1
1	0	0	1=z’	0=0
1	0	1	0=z’	0=0
1	1	0	0=z	0=z
1	1	1	1=z	1=z





PART 2: DIGITAL DESIGN LAB (TASK1) (CHECKED)

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.*

CONSTRAINT FILE AND THE TESTING

Describe how you test your design on the Minisys Practice platform.

- *Constraint file (provide the screen shots on the feature of a pin and the binding info between pins and the input /output ports)*
- *The testing result (provide the screen shots (at least 3 testing scene) to show state of inputs and outputs along with the related descriptions.*

THE DESCRIPTION OF OPERATION

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

- *Problems and solutions*

PART 2: DIGITAL DESIGN LAB (TASK2) (CHECKED)

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)*
- *Block design (provide screen shots)*
- *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*

CONSTRAINT FILE AND THE TESTING

Describe how you test your design on the Minisys Practice platform.

- *Constraint file (provide the screen shots on the feature of a pin and the binding info between pins and the input /output ports)*
- *The testing result (provide the screen shots (at least 3 testing scene)) to show state of inputs and outputs along with the related descriptions.*

THE DESCRIPTION OF OPERATION

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

- *Problems and solutions*