EE2026
Tutorial 3 - Solutions

Logic gates

1. $F=$

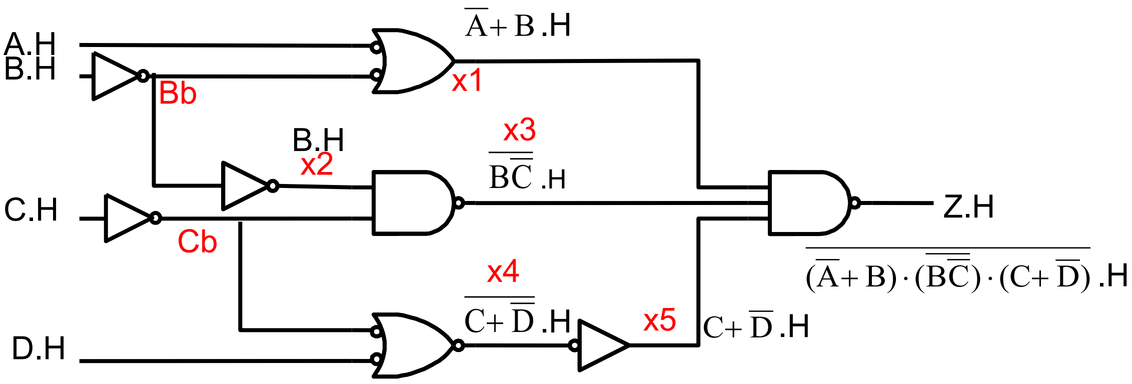
$$
\begin{aligned}
&x_1x_3+x_1\bar{x}_2+\bar{x}_1x_2x_3+\bar{x}_1\bar{x}_2\bar{x}_3\\
&=x_3(x_1+\bar{x}_1x_2)+\bar{x}_2(x_1+\bar{x}_1\bar{x}_3)\\
&=x_3(x_1+x_2)+\bar{x}_2(x_1+\bar{x}_3)\ \{\text{using } A+\bar{A}B=A+B\}\\
&=x_1x_3+x_2x_3+x_1\bar{x}_2+\bar{x}_2\bar{x}_3\\
&=x_1x_3+x_2x_3+\bar{x}_2\bar{x}_3\quad \{\text{using } AB+\bar{A}C+BC=AB+\bar{A}C;\ A\rightarrow x_3,\ B\rightarrow x_1,\ C\rightarrow \bar{x}_2\}\\
&\text{or}\ \ x_1\bar{x}_2+x_2x_3+\bar{x}_2\bar{x}_3\quad \{\text{using } AB+\bar{A}C+BC=AB+\bar{A}C;\ A\rightarrow \bar{x}_2,\ B\rightarrow x_1,\ C\rightarrow x_3\}
\end{aligned}
$$

The Verilog module of the original function is as follows (in practical cases, the simplified version will be derived automatically by the synthesis tool, as will be shown in lab sessions):

```
module func(x1,x2,x3,F);
        input x1,x2,x3;
        output F;
        assign F = x1 & x3 | x1 & ~x2 | ~x1 & x2 & x3 | ~x1 & ~x2 & ~x3;
        // no parentheses are needed
endmodule
```

2. Express all inputs and outputs as active high (i.e., complement active-low signals). The resulting circuit is in positive logic, and the expression of all intermediate nodes x1...x5 is found immediately by proceeding from inputs to output:

Circuit 1

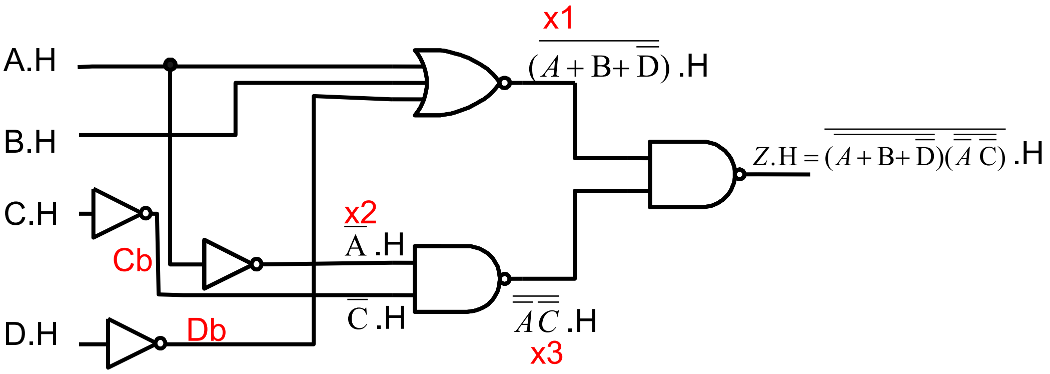


A.H
B.H
Bb
$\bar{A}+B$ .H
x1
B.H
x2
x3
$\overline{B\bar{C}}$ .H
C.H
Cb
Z.H
$\overline{(\bar{A}+B)\cdot(\overline{B\bar{C}})\cdot(C+\bar{D})}$ .H
x4
$\overline{C+\bar{D}}$ .H
x5
$C+\bar{D}$ .H
D.H

From inspection of the above gate-level structure, the Verilog structural description is:

```
module func(Z,A,B,C,D); // Bb and Cb are complemented (active-low)
        input A, B, C, D;
        output Z;
        wire x1, x2, x3, x4, x5, Bb, Cb; // need to define intermediate wires
                not u1(Bb,B); // inverter gate to generate Bb=NOT(B) (omitted inverter)
                not u2(Cb,C); // inverter gate to generate Cb=NOT(C) (omitted inverter)
                nand u3(x1,A,Bb); // NAND2 gate, instance u3 (push bubbles in gate @top-left)
                not u4(x2,Bb); // inverter gate
                nand u5(x3,x2,Cb); // NAND2 gate
                and u6(x4,Cb,d); // AND2 gate (push bubbles in gate at bottom-left)
                not u7(x5,x4); // inverter gate
                nand u8(Z,x1, x3,x5); // NAND3 gate
endmodule
```

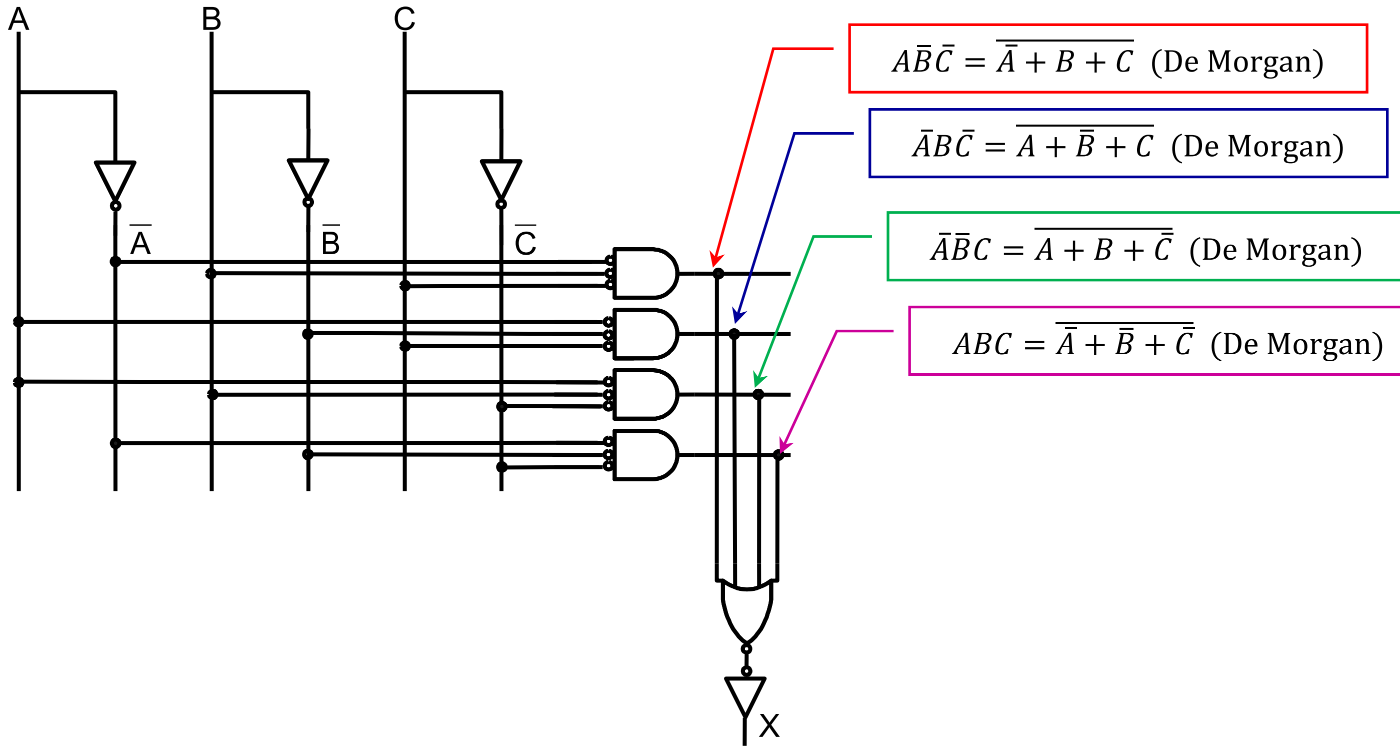
Circuit 2


x1
$\overline{(A+B+\overline{D})}$ .H
A.H
B.H
C.H
D.H
$Z.H = \overline{\overline{(A+B+\overline{D})}\,\overline{(\overline{A}\,\overline{C})}}$ .H
Cb
Db
x2
$\overline{A}$ .H
$\overline{C}$ .H
$\overline{\overline{A}\,\overline{C}}$ .H
x3

From inspection of the above gate-level structure, the Verilog structural description is:

```
module func(Z,A,B,C,D); // Cb and Db are complemented (active-low)
        input A, B, C, D;
        output Z;
        wire x1, x2, x3, Cb, Db;
                not u1(Cb,C); // inverter gate to generate Cb=NOT(C) (see omitted inverter)
                not u2(Db,D); // inverter gate to generate Db=NOT(D) (see omitted inverter)
                nor u3(x1,A,B,Db); // NOR3 gate, instance u3
                not u4(x2,A); // inverter gate
                nand u5(x3,x2,Cb); // NAND2 gate
                nand u6(Z,x1,x3); // NAND2 gate
endmodule
```

3. $X = A \oplus B \oplus C = (A\overline{B} + \overline{A}B) \cdot \overline{C} + \overline{A\overline{B} + \overline{A}B} \cdot C = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C + ABC$



$A\bar{B}\bar{C} = \overline{\bar{A} + B + C}$ (De Morgan)

$\bar{A}B\bar{C} = \overline{A + \bar{B} + C}$ (De Morgan)

$\bar{A}\bar{B}C = \overline{A + B + \bar{C}}$ (De Morgan)

$ABC = \overline{\bar{A} + \bar{B} + \bar{C}}$ (De Morgan)

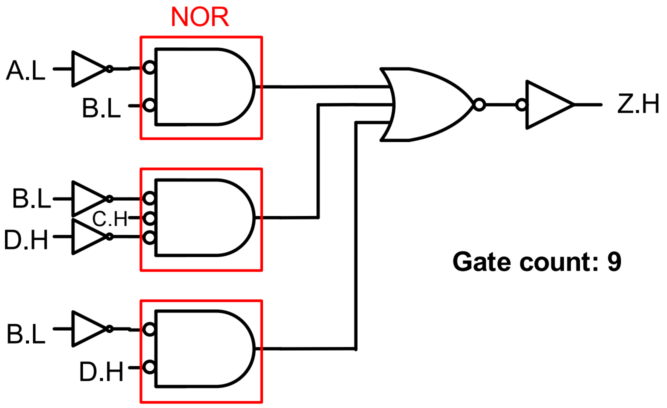
The dataflow Verilog description of the above function $X = A \oplus B \oplus C$ is:

```
module function(A,B,C,X);
        input A,B,C;
        output X;
        assign X = A ^ B ^ C; //
        // equivalent to A & ~B & ~C  | ~A & B & ~C | ~A & ~B & C | A & B & C (see above)
endmodule
```

Can you write the structural Verilog description, based on the above gate-level structure?

4. $\mathbf{Z} = \overline{\mathbf{A}}\,\mathbf{B} + \overline{\mathbf{B}}\,\overline{\mathbf{C}}\,\mathbf{D} + \overline{\mathbf{B}}\,\overline{\mathbf{D}}$ (A, B are active low)

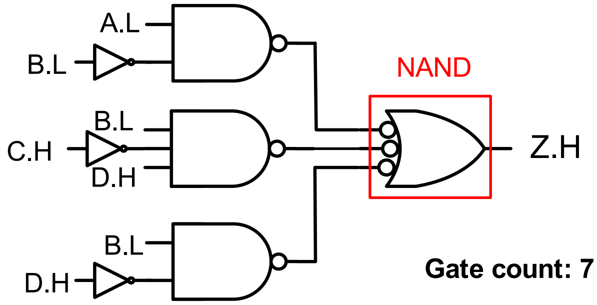
Using NOR gate:



Gate count: 9

All NOR gates + Not gates (inverters)

Using NAND gate:



Gate count: 7

All NAND gates + Not gates (inverters)

The dataflow Verilog description of the original function is:

```
module func(Z,A,B,C,D);
        input A, B, C, D;
        output Z;
        assign Z = A & ~B  | B & ~C & D | B & ~D; // A, B were complemented (active low)
endmodule
```

Can you describe it in a structural style, starting from the gate-level implementation shown above?