《数字逻辑》

（第6次实验：在Logisim和FPGA开发板上实现中规模组合逻辑电路）

厦门大学信息学院软件工程系　曾文华

2024年11月18日

目录

第一部分：在Logisim上实现中规模组合逻辑电路

请打开设计文件“第7章例题电路的实现（第1部分）.circ”

（一）在Logisim上实现教材第7章的例题

- **1、（验证实验）4位二进制串行加法器**
  - 在Logisim上实现4位二进制串行加法器（图7.1）。
  - 验证步骤：
    - 按Ctrl+R，设置不同的A、B、C0值，观看结果F、C4是否正确?

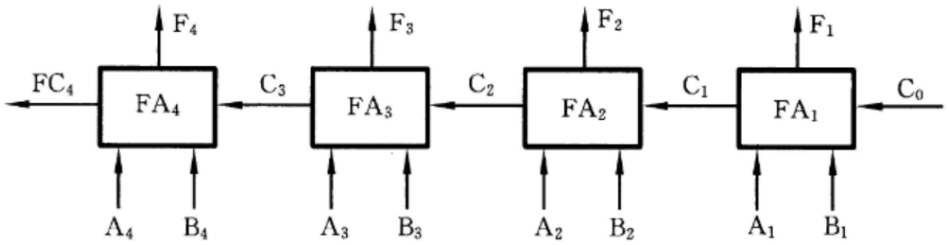
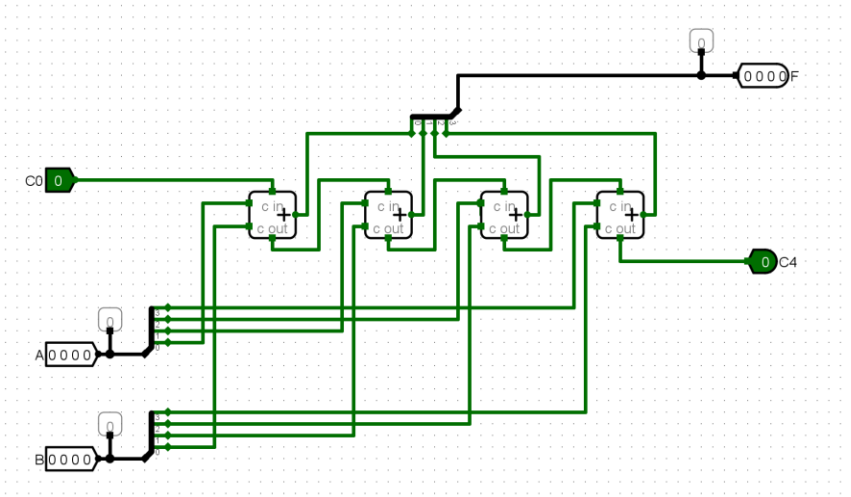


图 7.1　4 位串行进位二进制并行加法器的结构框图



• 2、（验证实验）4位二进制并行加法器（超前进位）

- 在Logisim上验证4位二进制并行（超前进位）加法器74283芯片（图7.2）。

- **验证步骤：**
  - 按Ctrl+R，设置不同的A、B、C0值，观看结果F、C4是否正确?

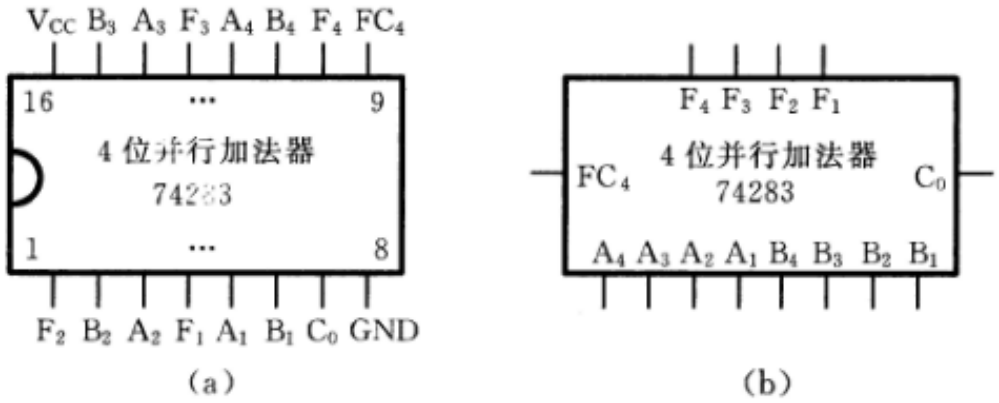
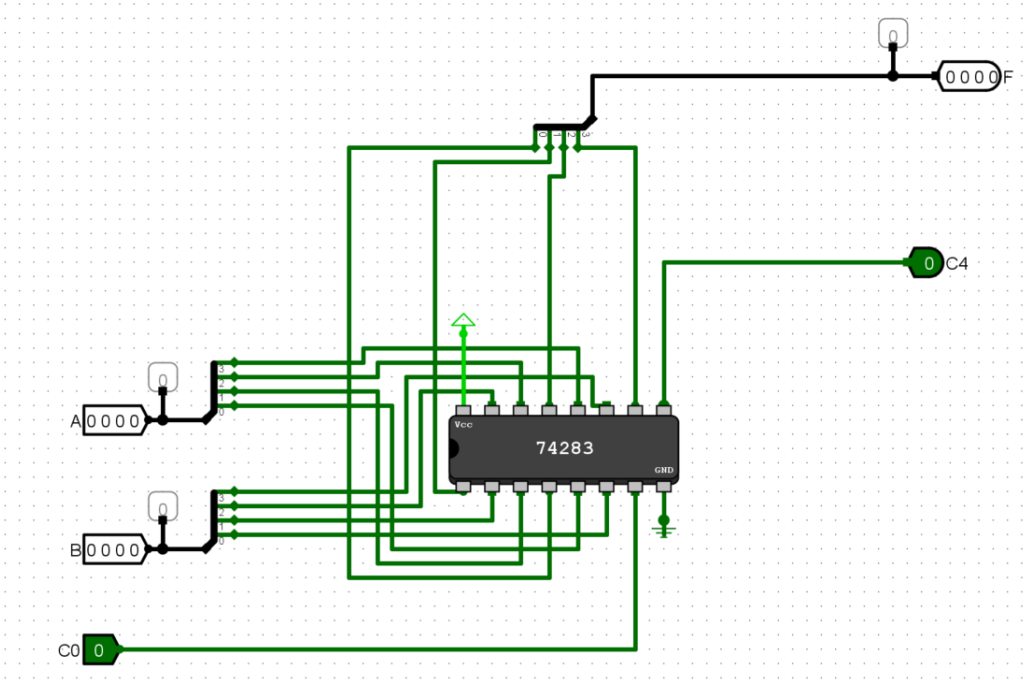


图 7.2　74283 的引脚排列图和逻辑符号



• 3、（验证实验）例题7.1的实现

- 在Logisim上实现例题7.1的8421码转换成余3码的代码转换电路（图7.3，74283芯片为4位二进制并行（超前进位）加法器）。

- **验证步骤：**
  - 先按复位按钮RST，再按时钟按钮CLK多次，观看输入的8421码是否正确地转换为余3码（表1.3）？

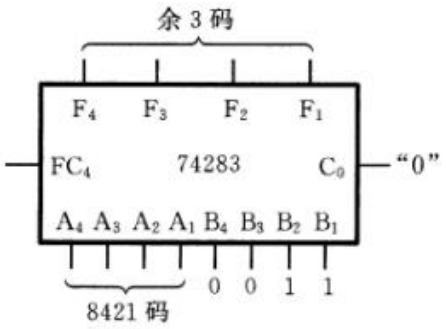
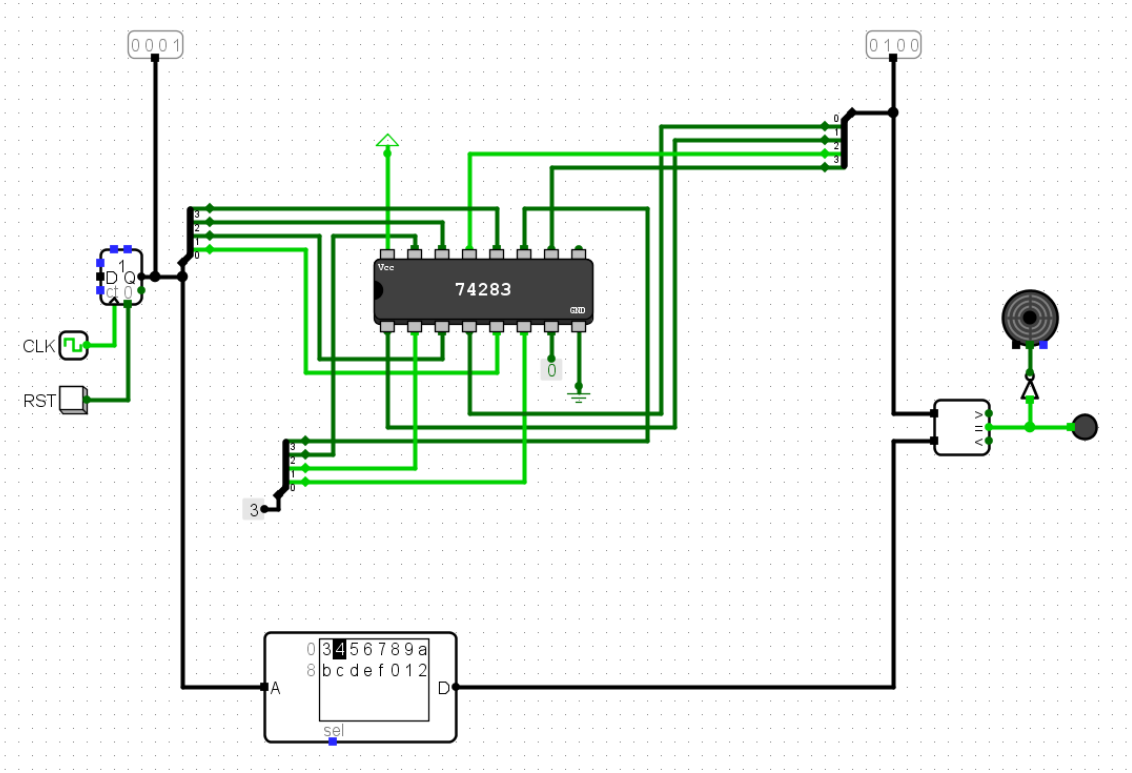


图 7.3　逻辑电路

表 1.3　常用的 3 种 BCD 码

十进制字符	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100



- 4、（验证实验）例题7.2的实现
  - 在Logisim上实现例题7.2的4位二进制并行加法/减法器（图7.4，74283芯片为4位二进制并行（超前进位）加法器）。
  - 验证步骤：
    - 按Ctrl+R，置M=0，设置不同的被加数A和加数B，观看和（加法结果）F是否正确？置M=1，设置不同的被减数A和减数B，观看差（减法结果）F是否正确？

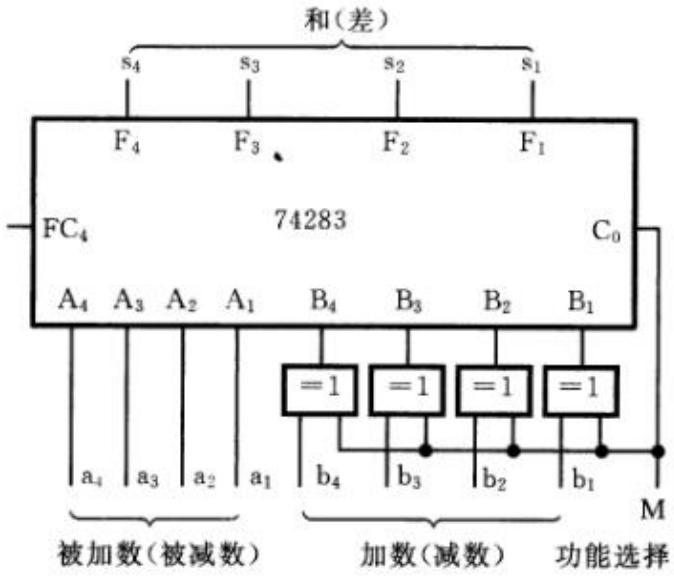
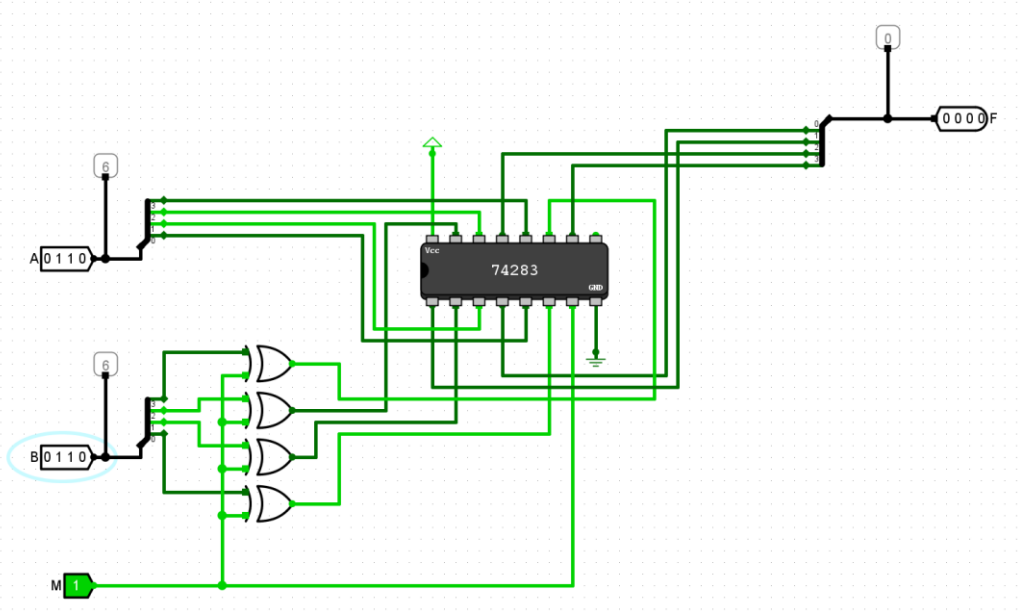


图 7.4 逻辑电路



• 5、（验证实验）例题7.3的实现

- 在Logisim上实现例题7.3的用余3码表示的1位十进制数加法器（图7.5，74283芯片为4位二进制并行（超前进位）加法器）。

- 验证步骤：
  - 按Ctrl+R，置Cin=0，设置不同的被加数余3码A和加数余3码B，观看和数（加法结果）余3码F和Cout是否正确（表1.3）？置Cin=1，设置不同的被加数余3码A和加数余3码B，观看和数（加法结果）余3码F和Cout是否正确（表1.3）？

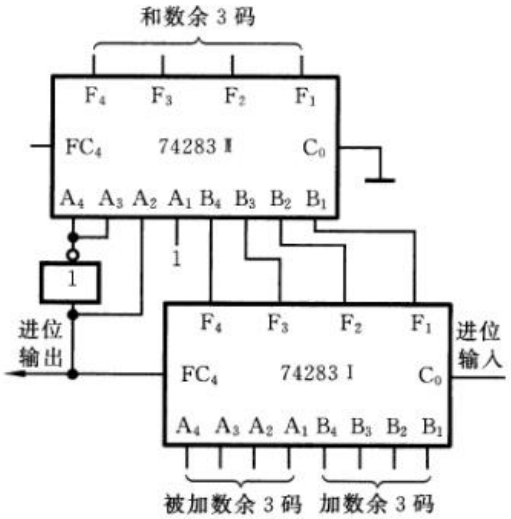
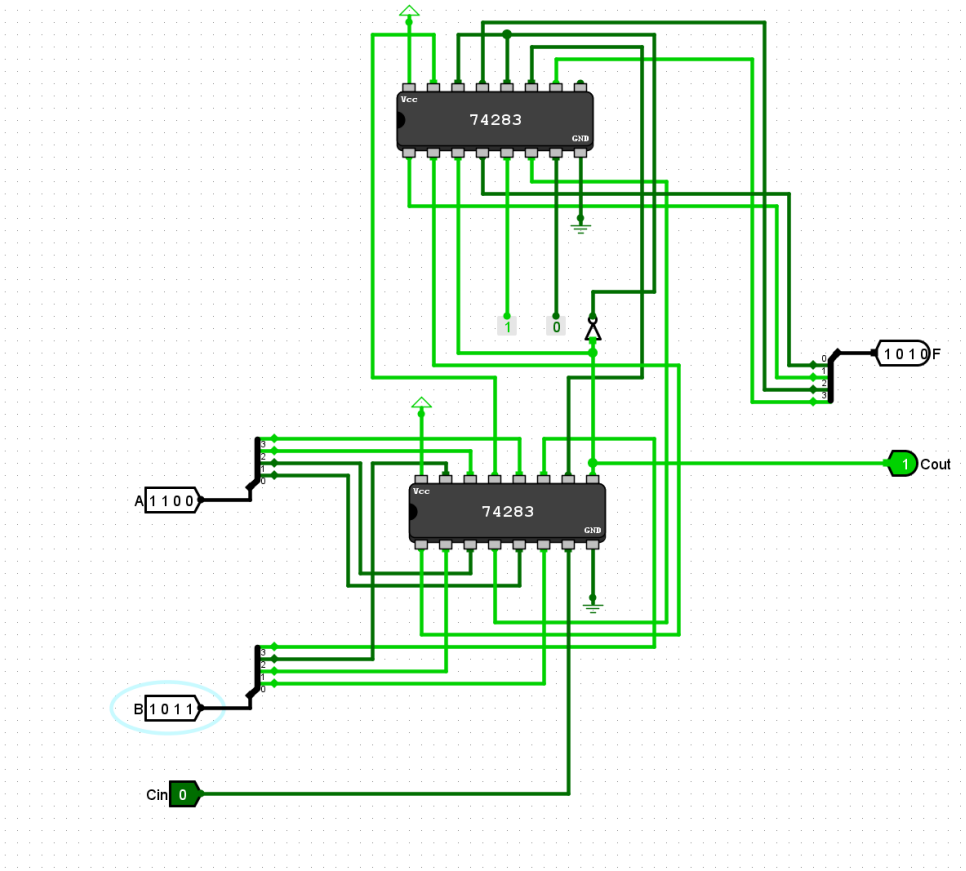


图 7.5　逻辑电路

表 1.3　常用的 3 种 BCD 码

十进制字符	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100



• 6、（验证实验）例题7.4的实现

- 在Logisim上实现例题7.4的4位二进制乘法器（图7.6，74283芯片为4位二进制并行（超前进位）加法器）。

- 验证步骤：
  - 按Ctrl+R，设置不同的乘数A和被乘数B，观看乘积F是否正确？

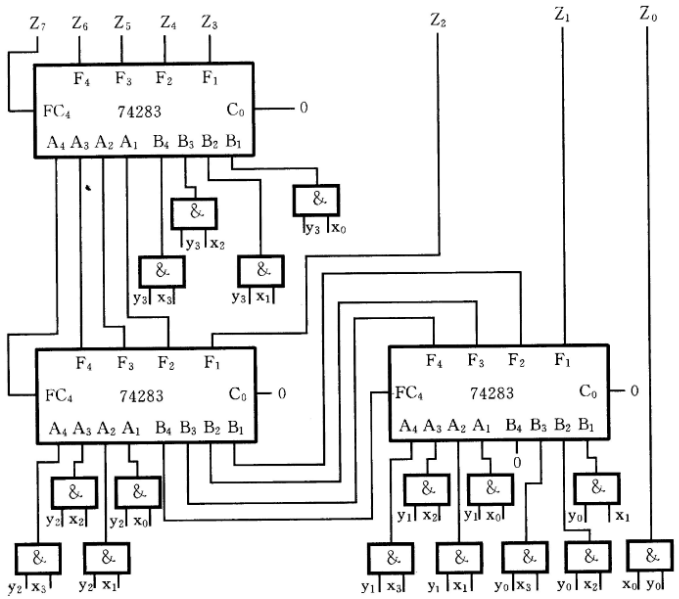
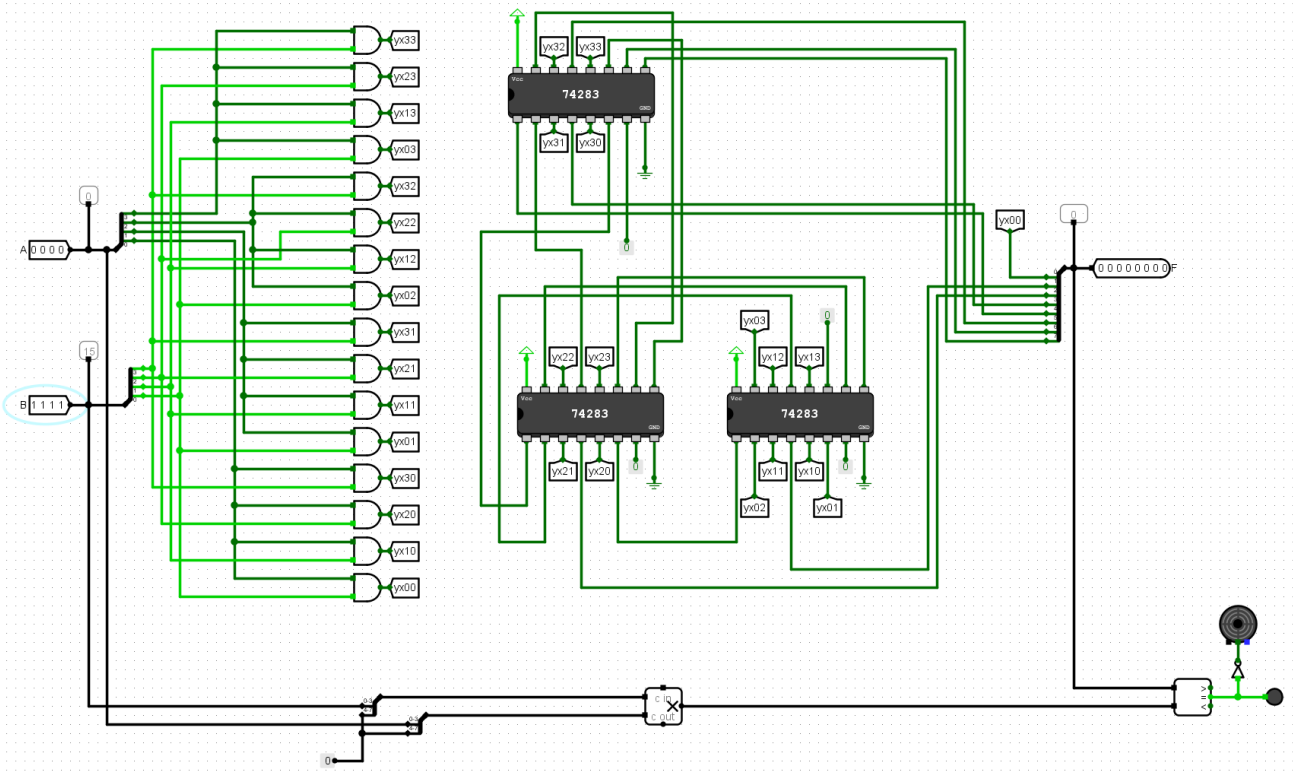


图 7.6　逻辑电路



• 7、（验证实验）各类译码器

- 在Logisim上实现1-2译码器、2-4译码器、3-8译码器、4-16译码器，以及3-8译码器74138芯片（图7.7）。

- **验证步骤**：
  - 按Ctrl+R，验证3-8译码器74138芯片的真值表（表7.1）是否正确？

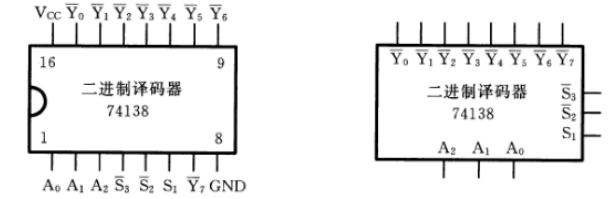
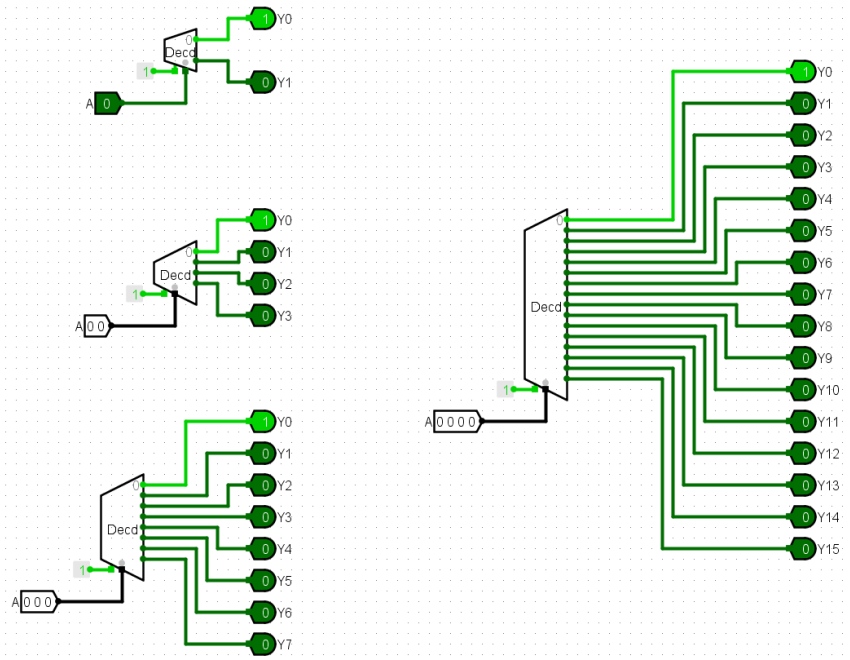
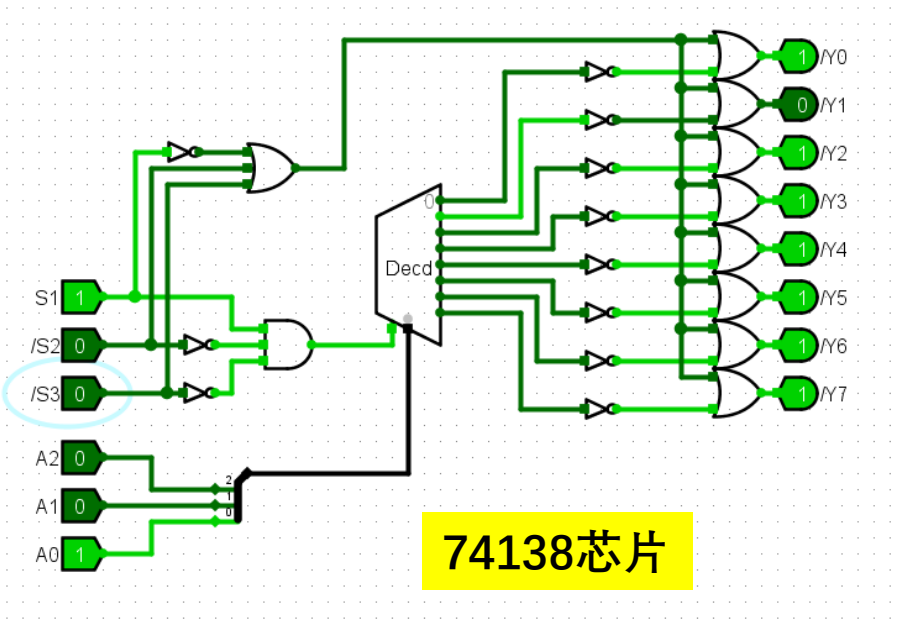


图 7.7　74138 译码器的引脚排列图和逻辑符号

表 7.1　74138 译码器真值表

输入					输出							
S_1	$\overline{S}_2+\overline{S}_3$	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	1	1	1	1	1	1	1	1
d	1	d	d	d	1	1	1	1	1	1	1	1





• 8、（验证实验）例题7.5的实现

- 在Logisim上实现例题7.5的全减器（图7.8，74138为3-8译码器芯片）。

- 验证步骤：
  - 按Ctrl+R，然后设置不同的A_i、B_i、G_{i-1}，观看D_i和G_i是否正确（表7.2）？

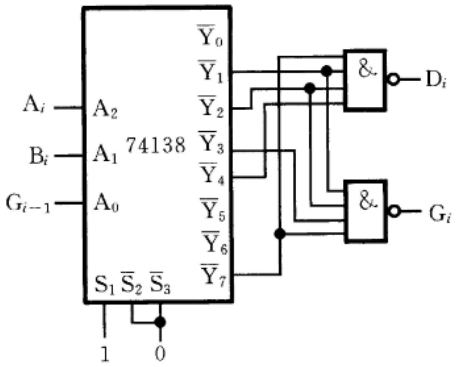
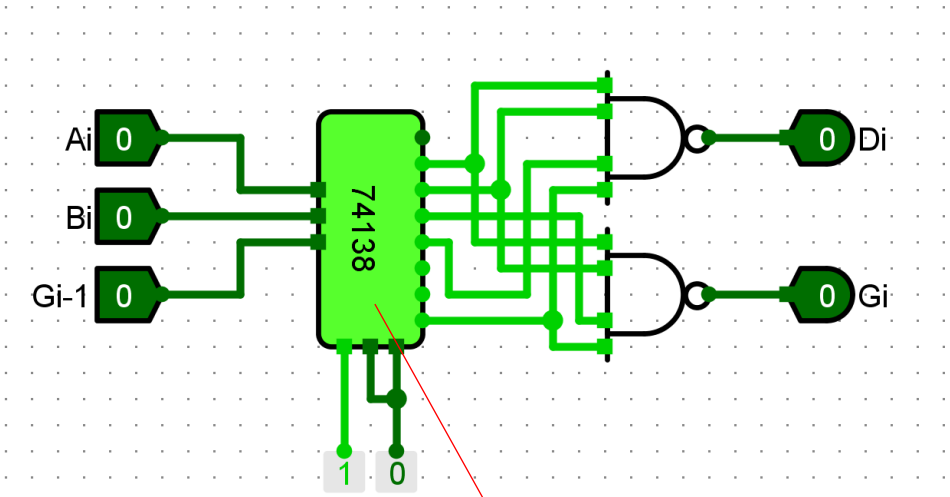
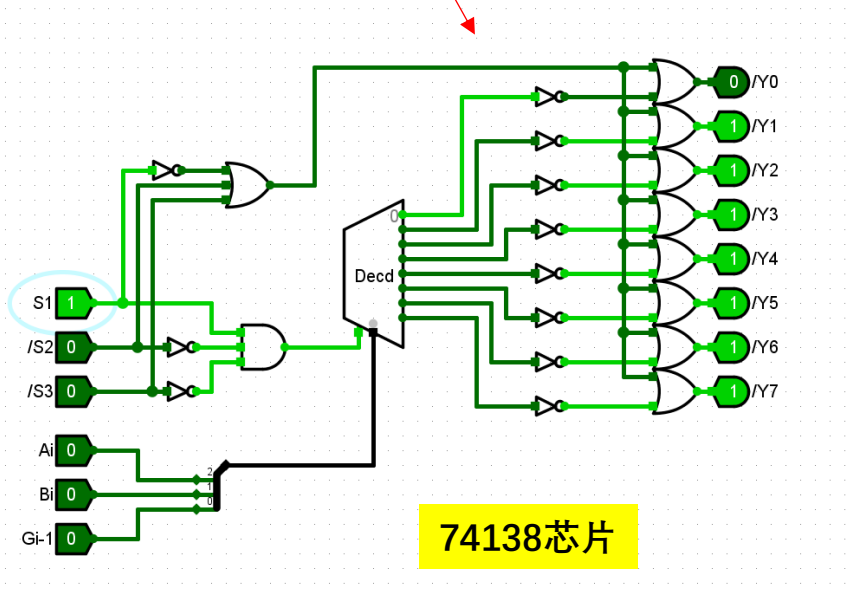


图 7.8　逻辑电路

表 7.2　全减器真值表

输入			输出	
A_i	B_i	G_{i-1}	D_i	G_i
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





• 9、（验证实验）例题7.6的实现

- 在Logisim上实现例题7.6的逻辑电路（图7.9，74138为3-8译码器芯片）：F(A,B,C,D)= ∑m(2,4,6,8,10,12,14)。

- 验证步骤：
  - 先按复位按钮RST，再按时钟按钮CLK多次，观看是否得到正确的结果（输入为2、4、6、8、10、12、14时，输出=1；其余情况，输出=0）？

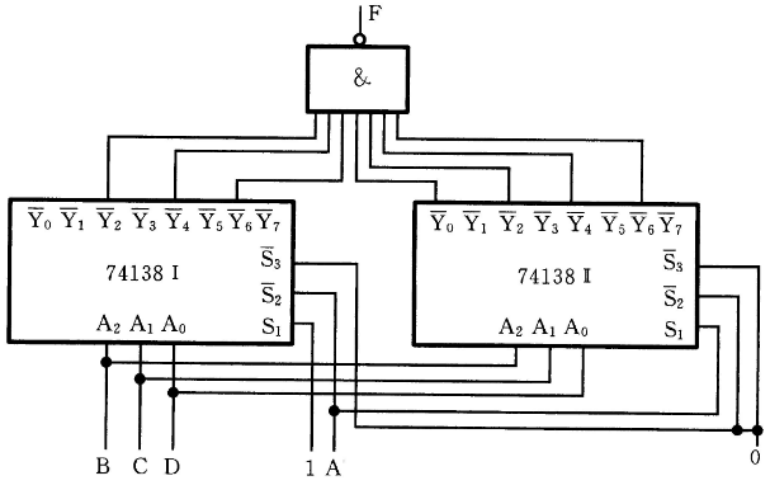
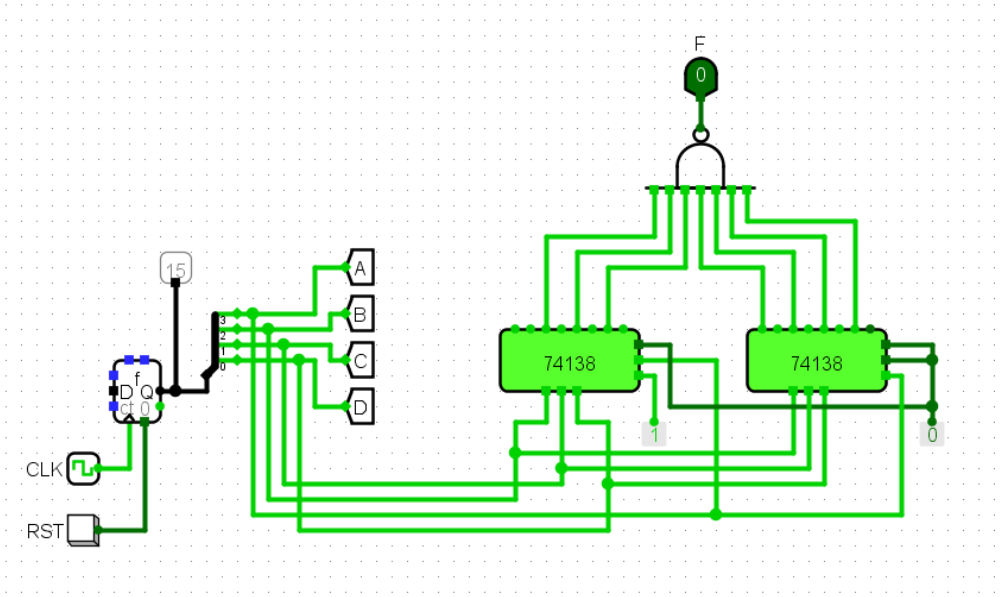


图 7.9　逻辑电路



• 10、（验证实验）二-十进制译码器

- 在Logisim上实现二-十进制译码器7442芯片（图7.10）。

- **验证步骤：**
  - 先按复位按钮RST，再按时钟按钮CLK多次，观看是否得到正确的结果（表7.3）？

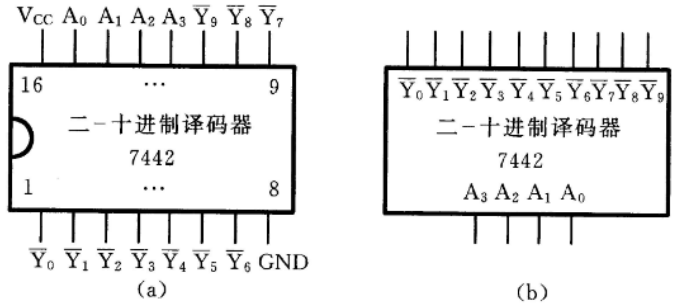
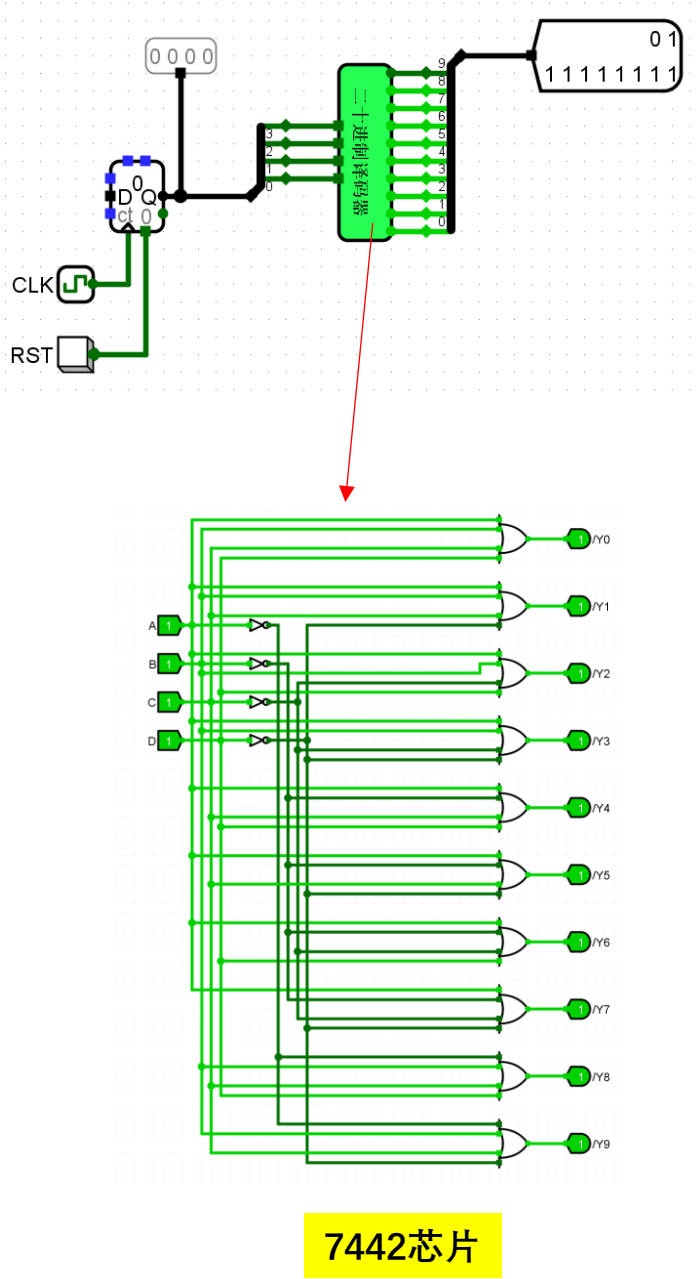


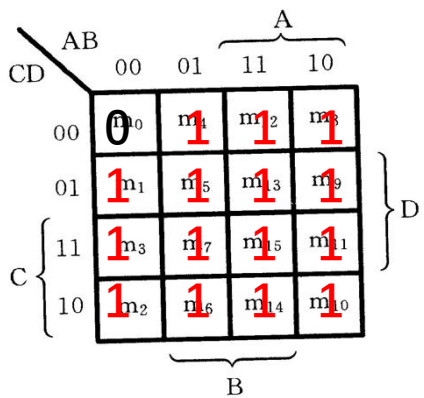
图 7.10　二-十进制译码器 7442 的引脚排列图和逻辑符号

表 7.3　二-十进制译码器 7442 的真值表

输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



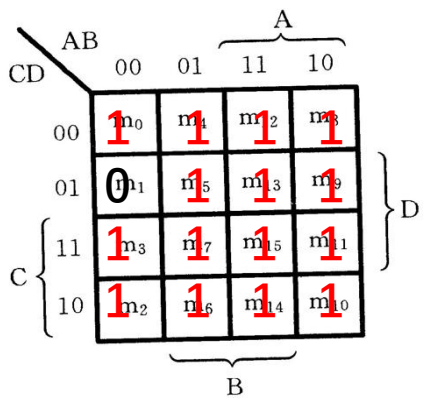
7442芯片



$/Y_0=C+D+A+B$

表 7.3　二-十进制译码器 7442 的真值表

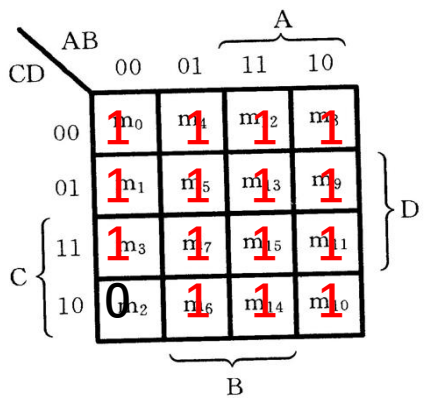
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_1=C+/D+A+B$

表 7.3　二-十进制译码器 7442 的真值表

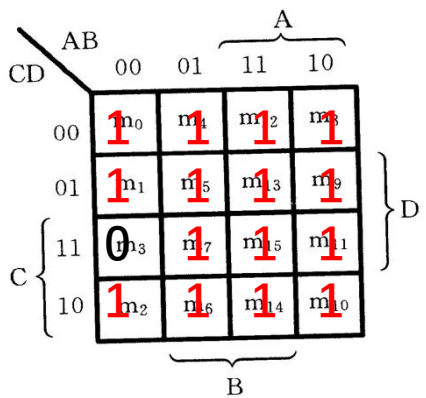
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_2=/C+D+A+B$

表 7.3　二-十进制译码器 7442 的真值表

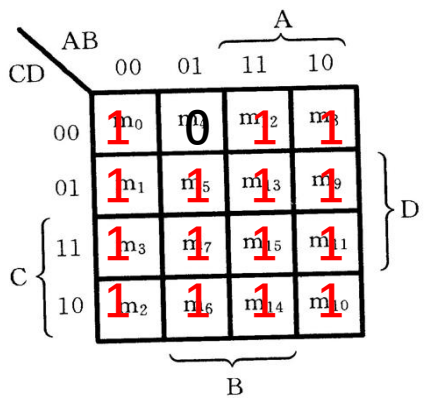
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_3=/C+/D+A+B$

表 7.3　二-十进制译码器 7442 的真值表

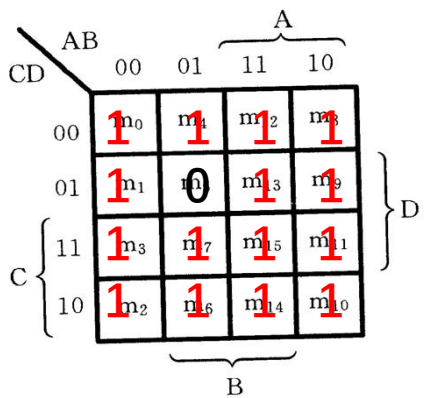
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$$/Y_4=C+D+A+/B$$

表 7.3　二-十进制译码器 7442 的真值表

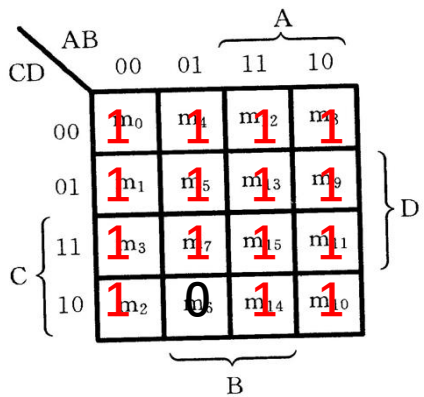
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_5=C+/D+A+/B$

表 7.3　二-十进制译码器 7442 的真值表

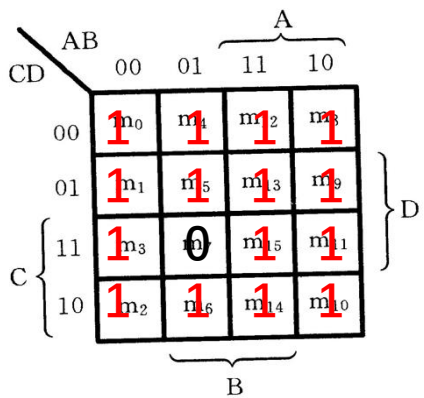
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_6=/C+D+A+/B$

表 7.3　二-十进制译码器 7442 的真值表

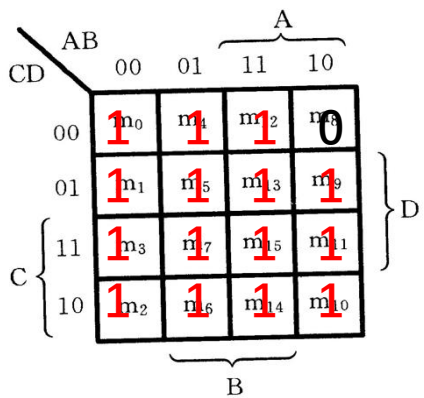
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_7=/C+/D+A+/B$

表 7.3　二-十进制译码器 7442 的真值表

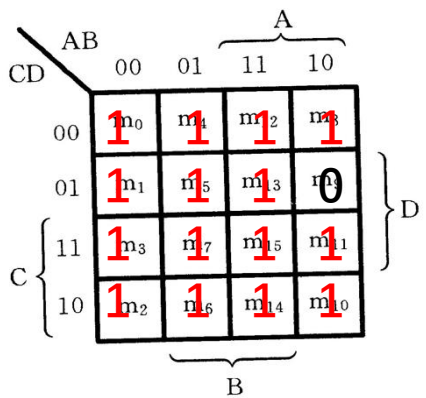
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_8=C+D+/A+B$

表 7.3　二-十进制译码器 7442 的真值表

输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



$/Y_9=C+/D+/A+B$

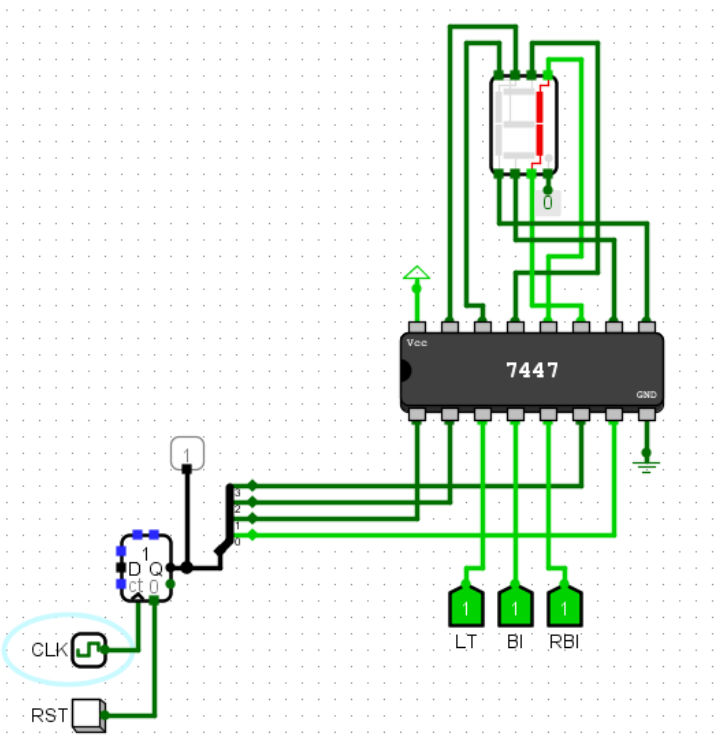
表 7.3　二-十进制译码器 7442 的真值表

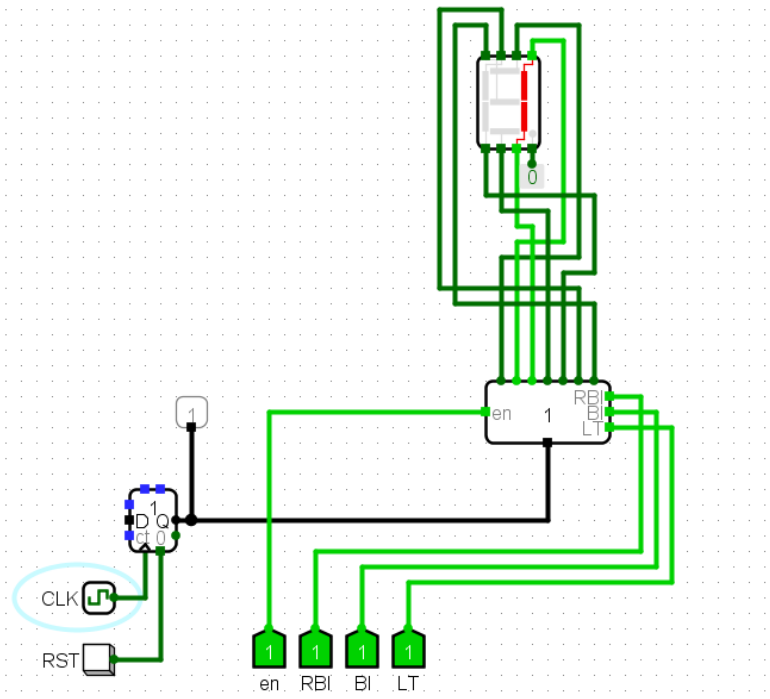
输入				输出									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

• 11、（验证实验）七段显示译码器

- 在Logisim上验证七段显示译码器（两种，其中一种为7447芯片。教材上为7448芯片）。

- 验证步骤：
  - （左下图）先按复位按钮RST，置LT=1，BI=1，RBI=1，再按时钟按钮CLK多次，观看数码管是否显示相应的数字?
  - （右下图）先按复位按钮RST，置en=1，RBI=1，BI=1，LT=1，再按时钟按钮CLK多次，观看数码管是否显示相应的数字?





• 12、（验证实验）二-十进制编码器（8421编码器）

- 在Logisim上实现二-十进制编码器（8421码编码器，图7.13）。

- **验证步骤：**
  - 先按Ctrl+R，然后按照表7.5设置不同的I_0～I_9，观看输出A、B、C、D、S是否正确？

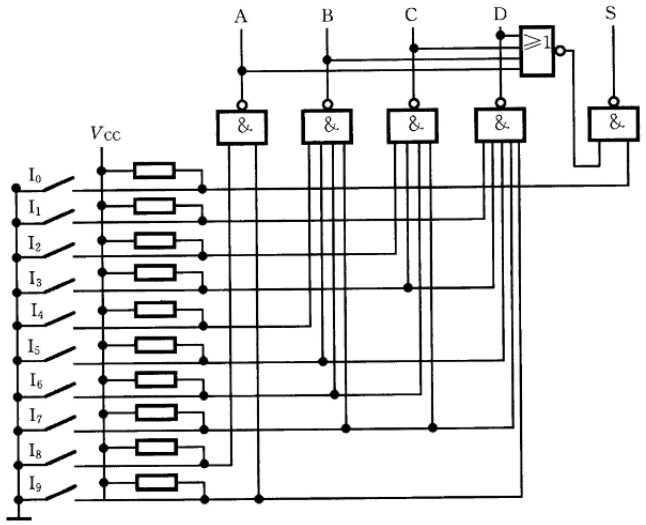
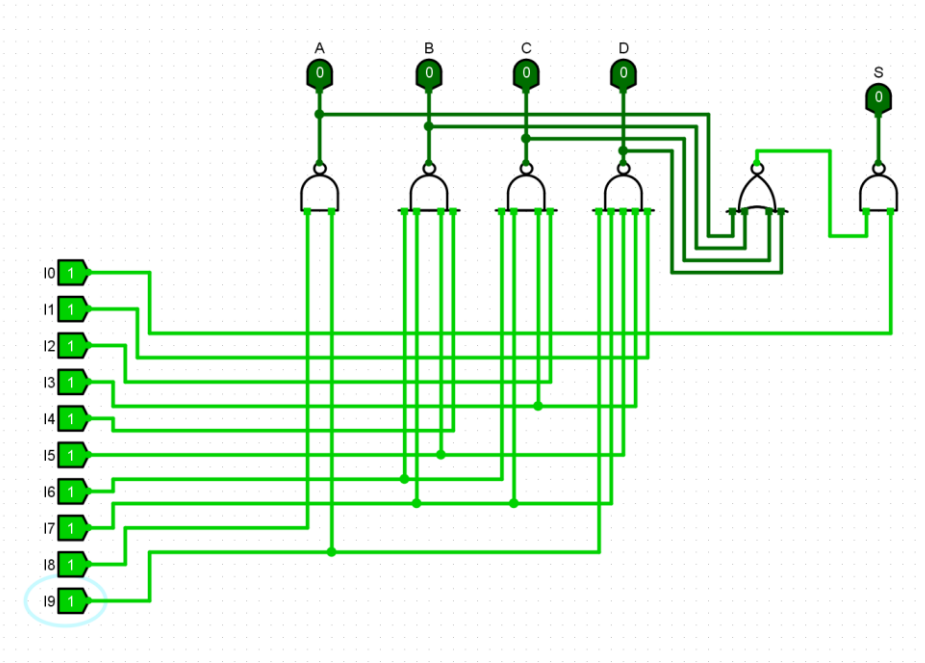


图 7.13　按键式 8421 码编码器

表 7.5　8421 码编码器真值表

输入										输出				
I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	A	B	C	D	S
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	1
1	1	1	1	1	1	1	1	0	1	0	0	0	1	1
1	1	1	1	1	1	1	0	1	1	0	0	1	0	1
1	1	1	1	1	1	0	1	1	1	0	0	1	1	1
1	1	1	1	1	0	1	1	1	1	0	1	0	0	1
1	1	1	1	0	1	1	1	1	1	0	1	0	1	1
1	1	1	0	1	1	1	1	1	1	0	1	1	0	1
1	1	0	1	1	1	1	1	1	1	0	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	0	0	0	1
0	1	1	1	1	1	1	1	1	1	1	0	0	1	1



• 13、（验证实验）优先编码器

- 在Logisim上实现8-3线优先编码器74148芯片（图7.14）。

- **验证步骤：**
  - 先按Ctrl+R，然后按照表7.6设置不同的/I_S、/I_0～/I_7，观看输出/Q_C、/Q_B、/Q_A、/Q_{EX}、O_S是否正确？

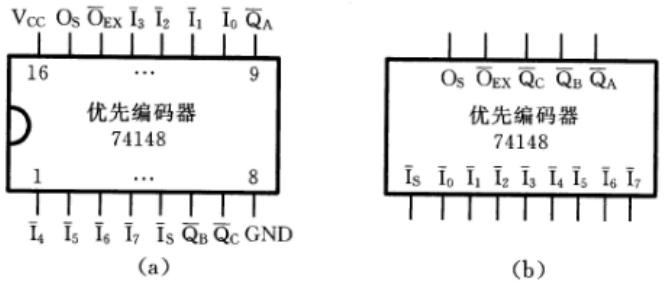
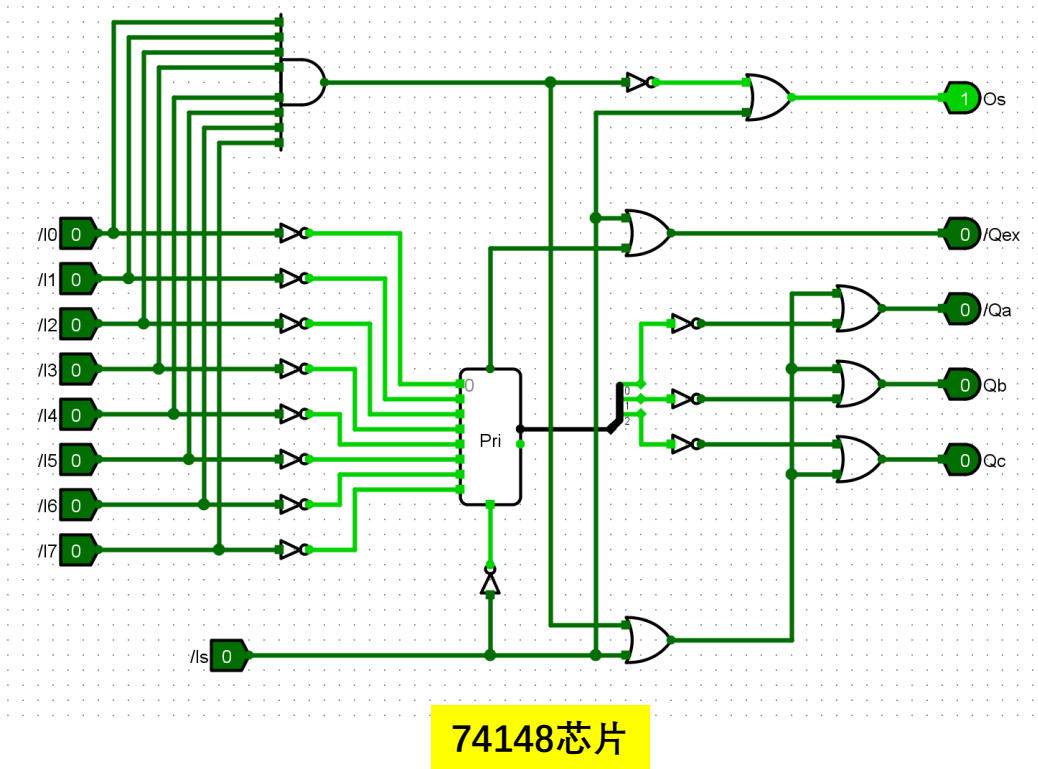


图 7.14　74148 优先编码器引脚排列和逻辑符号

表 7.6　74148 真值表

输入									输出				
$\overline{I}_S$	$\overline{I}_0$	$\overline{I}_1$	$\overline{I}_2$	$\overline{I}_3$	$\overline{I}_4$	$\overline{I}_5$	$\overline{I}_6$	$\overline{I}_7$	$\overline{Q}_C$	$\overline{Q}_B$	$\overline{Q}_A$	$\overline{Q}_{EX}$	O_S
1	d	d	d	d	d	d	d	d	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	d	d	d	0	0	0	0	0	1
0	d	d	d	d	d	d	0	1	0	0	1	0	1
0	d	d	d	d	d	0	1	1	0	1	0	0	1
0	d	d	d	d	0	1	1	1	0	1	1	0	1
0	d	d	d	0	1	1	1	1	1	0	0	0	1
0	d	d	0	1	1	1	1	1	1	0	1	0	1
0	d	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1



74148芯片

• 14、（验证实验）例题7.7的实现

- 在Logisim上实现例题7.7的16级中断优先编码器（图7.15，74148芯片为8-3线优先编码器）。

- **验证步骤：**
  - 先按Ctrl+R，然后设置不同的输入，观看输出是否正确?

若/I_{ZS}=1，则O_{ZS}=1，/Q_{ZA}～/Q_{ZD}=1111，/O_{ZEX}=1

若/I_{ZS}=0，/I_{Z15}～/I_{Z0}全部为1，则O_{ZS}=0，/Q_{ZA}～/Q_{ZD}=1111，/O_{ZEX}=1

若/I_{ZS}=0，/I_{Z10}=0（/I_{Z11}～I_{Z15}全部为1，/I_{Z0}～I_{Z9}任意），则O_{ZS}=1，/Q_{ZA}～/Q_{ZD}=1010，/O_{ZEX}=0

若/I_{ZS}=0，/I_{Z3}=0（/I_{Z4}～I_{Z15}全部为1，/I_{Z0}～I_{Z2}任意），则O_{ZS}=1，/Q_{ZA}～/Q_{ZD}=0011，/O_{ZEX}=0

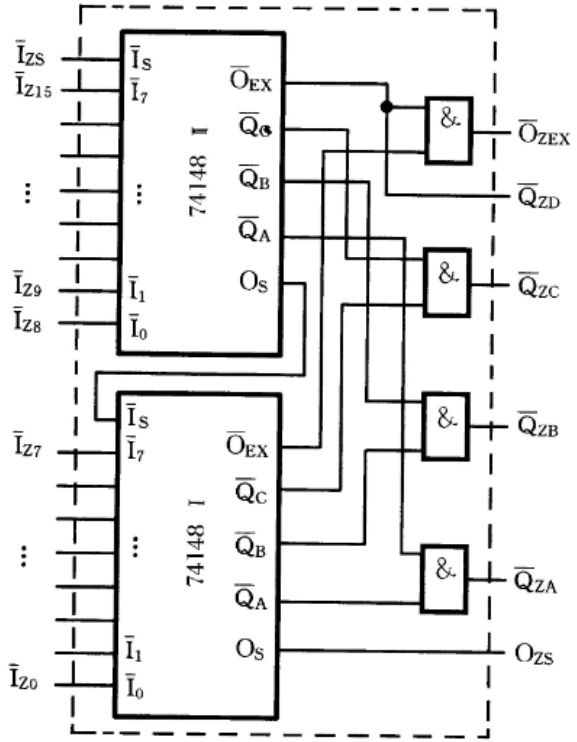
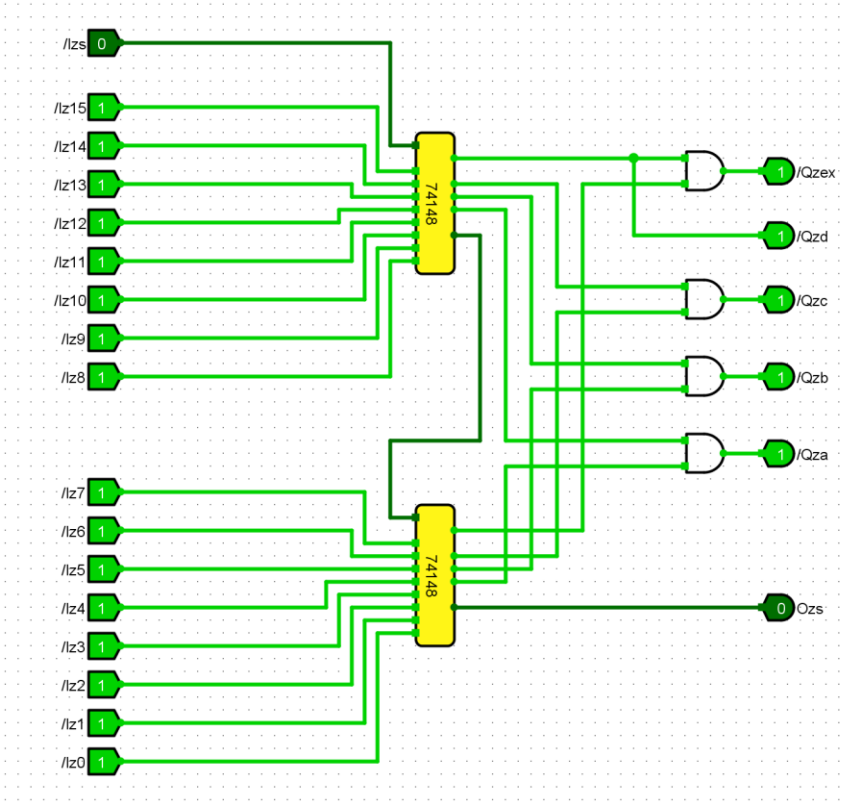


图 7.15　16 级中断优先编码器



• 15、（验证实验）例题7.8的实现

- 在Logisim上实现例题7.8的逻辑电路（两种方案，图7.17，74152芯片为8路多路选择器MUX，74153芯片为双4路多路选择器MUX）：Y(A,B,C)= ∑m(2,3,5,6)。

- 验证步骤：
  - （两个电路的验证步骤一致）先按复位按钮RST，再按时钟按钮CLK多次，观看输出是否正确（输入为2、3、5、6时，输出=1；其余情况，输出=0）？

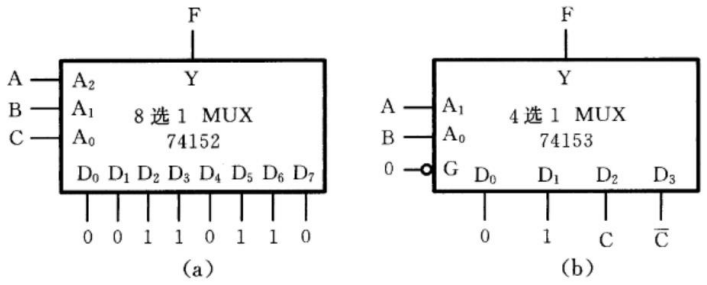
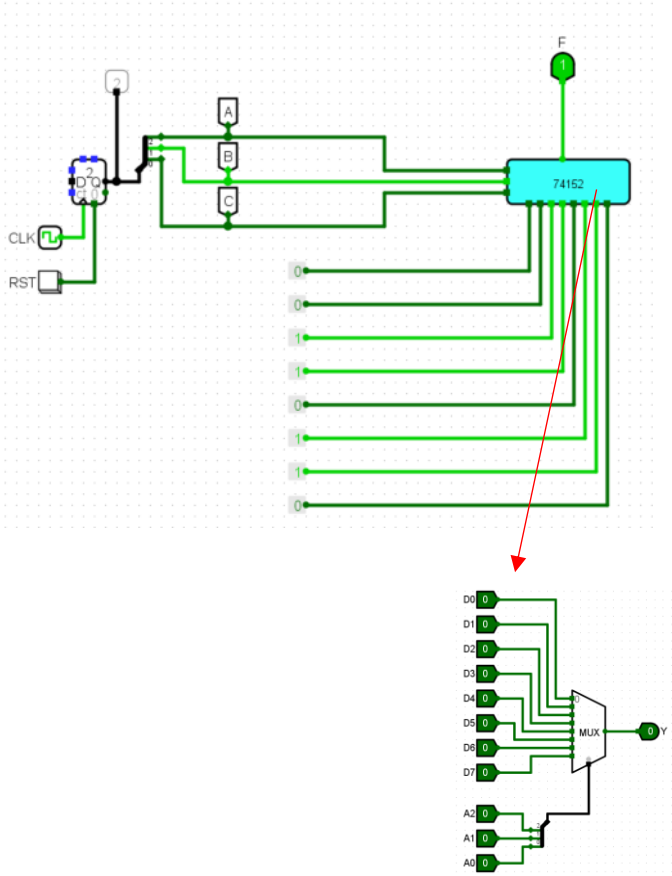
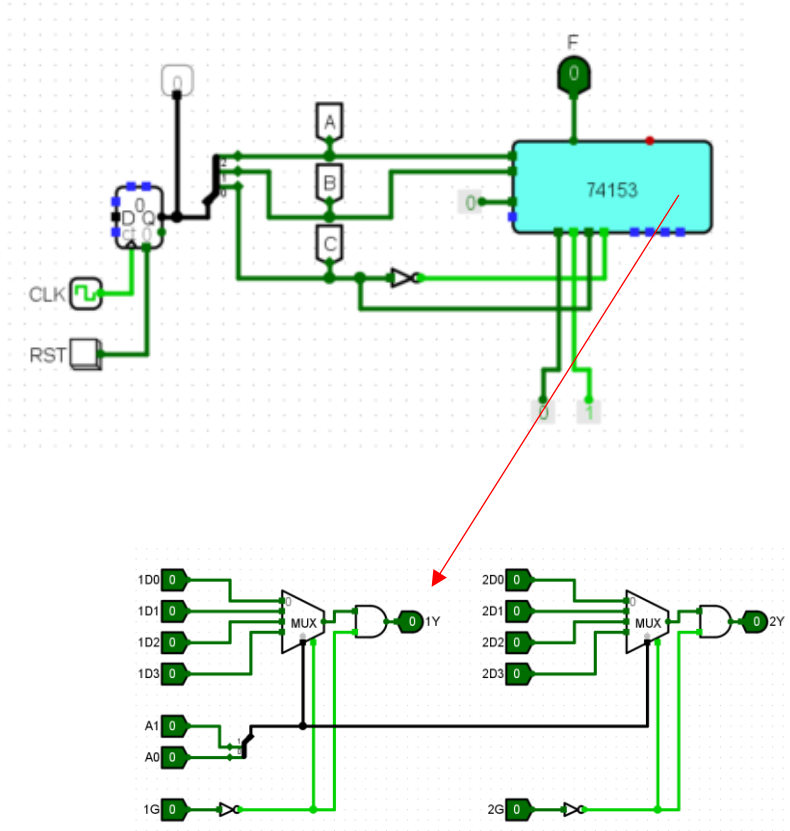


图 7.17　例 7.8 的两种方案





• 16、（验证实验）例题7.9的实现

- 在Logisim上实现例题7.9的逻辑电路（两种方案，图7.18，74153芯片为双4路多路选择器MUX）：$Y(A,B,C,D)=\sum m(0,2,3,7,8,9,10,13)$。

- 验证步骤：
  - （两个电路的验证步骤一致）先按复位按钮RST，再按时钟按钮CLK多次，观看输出是否正确（输入为0、2、3、7、8、9、10、13时，输出=1；其余情况，输出=0）。

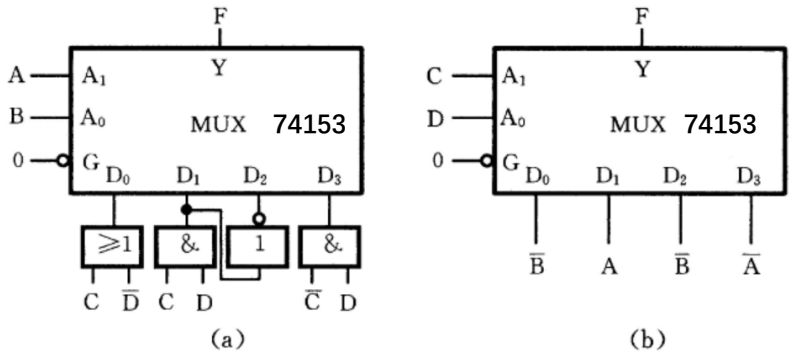
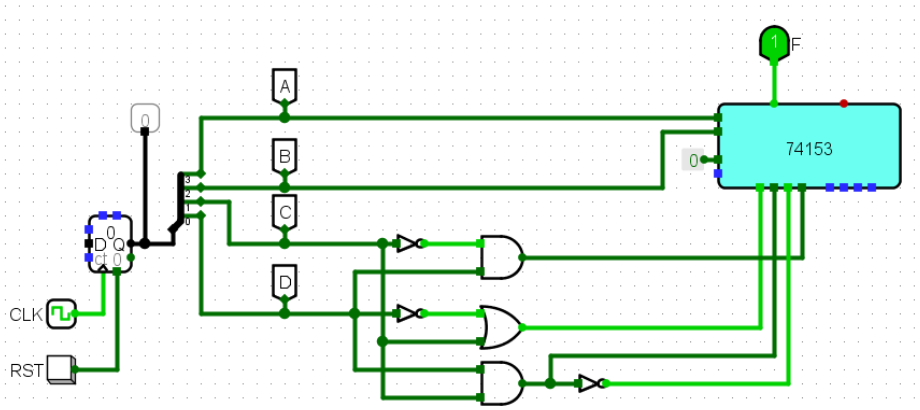
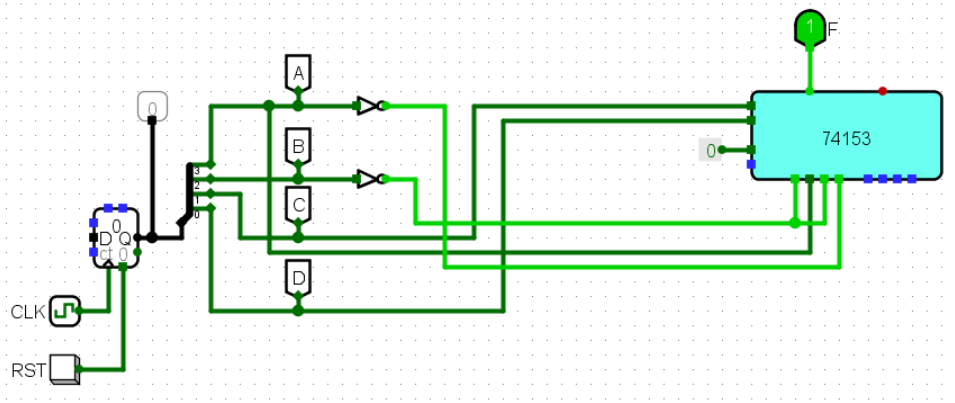


图 7.18　例 7.9 的两种方案





• 17、（验证实验）例题7.10的实现

- 在Logisim上实现例题7.9的逻辑电路（两个输出，图7.19，74153芯片为双4路多路选择器MUX）：$F_1(A,B,C,D)=\sum m(0,1,5,7,10,13,15)$，$F_2(A,B,C,D)=\sum m(8,10,12,13,15)$。

- 验证步骤：
  - 先按复位按钮RST，再按时钟按钮CLK多次，观看输出是否正确（输入为0、1、5、7、10、13、15时，F1输出=1；输入为8、10、12、13、15时，F2输出=1；其余情况，输出=0）？

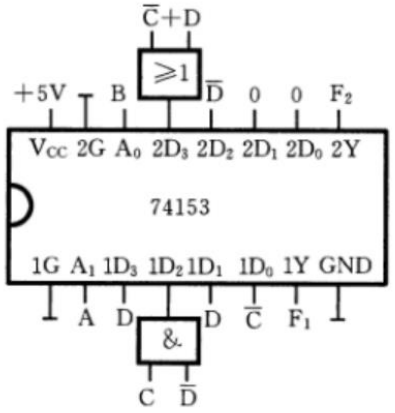
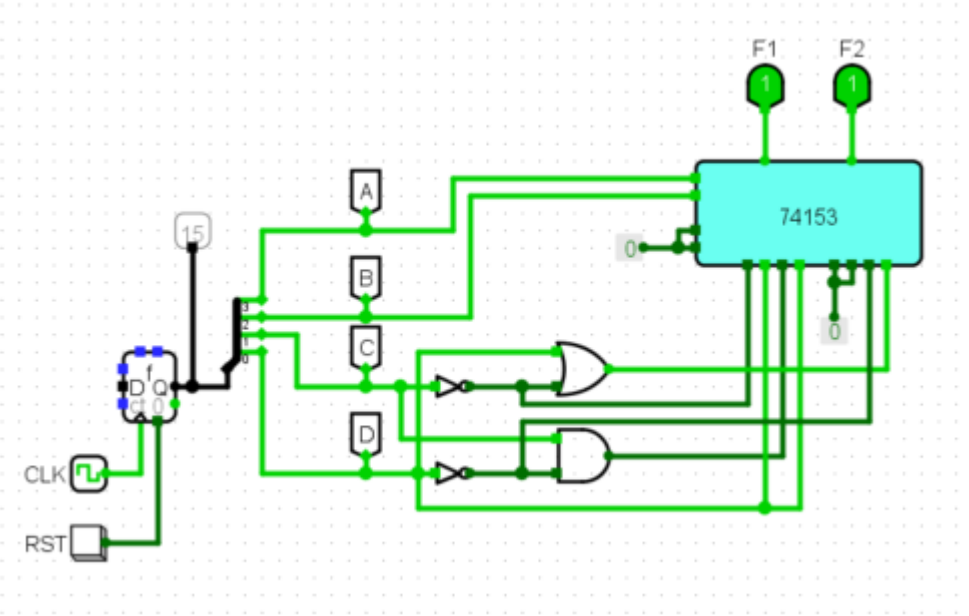


图 7.19　逻辑电路



- 18、（验证实验）多路分配器DEMUX
  - 在Logisim上实现4路分配器、8路分配器、8路数据传输（图7.21，74152芯片为8路多路选择器MUX）。
  - 验证步骤：
    - （8路数据传输）按Ctrl+R，设置不同的A、B、C值，点击与A、B、C值对应的输入（如ABC=110，则点击D6），观看相应的输出（f6）是否变化?

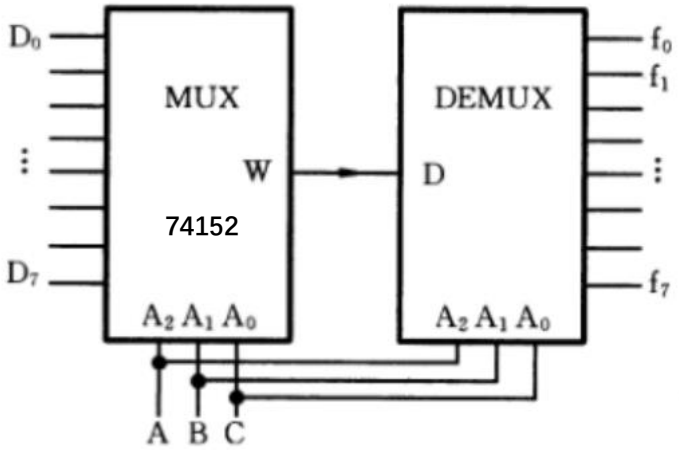
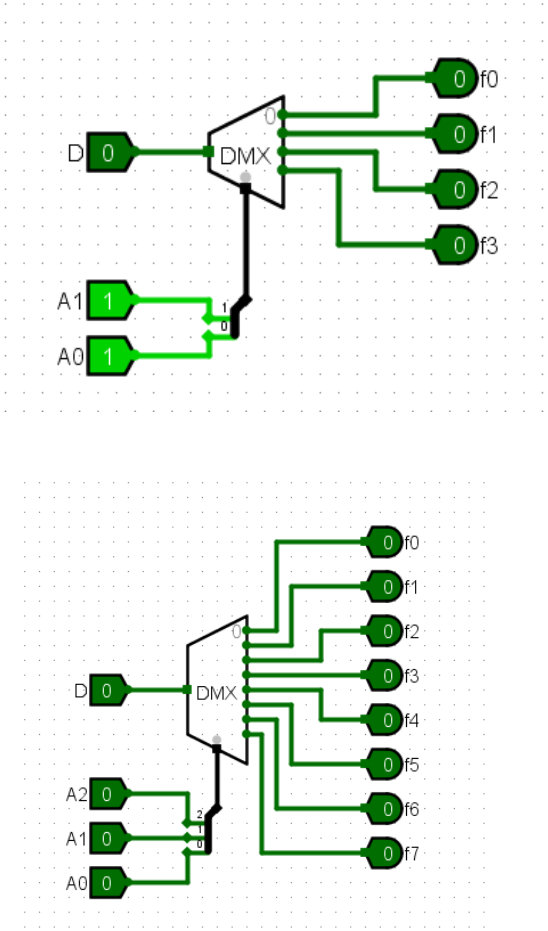
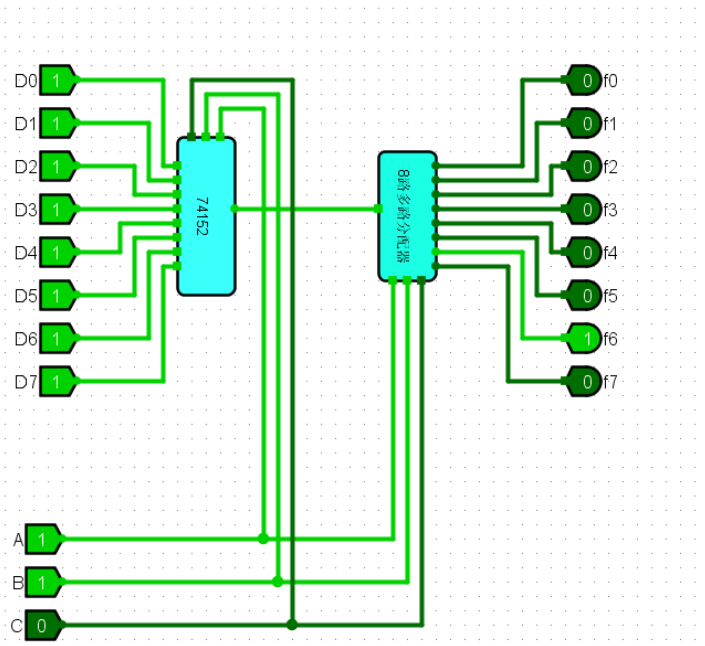


图 7.21　8 路数据传输示意图





• 19、（验证实验）例题7.11的实现

- 在Logisim上实现例题7.11的等值比较器（图7.22，74138芯片为3-8译码器，74152芯片为8路多路选择器MUX）。

- **验证步骤**：
  - 先按复位按钮RST，再按时钟按钮CLK多次，设置不同的A、B、C值，以及不同的X、Y、Z值，观看是不是“ABC=XYZ时，输出F=0； ABC≠XYZ时，输出F=1”？

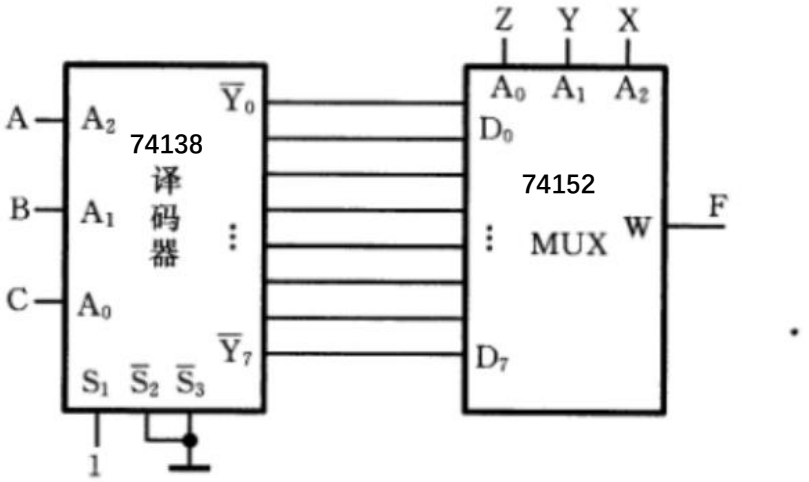
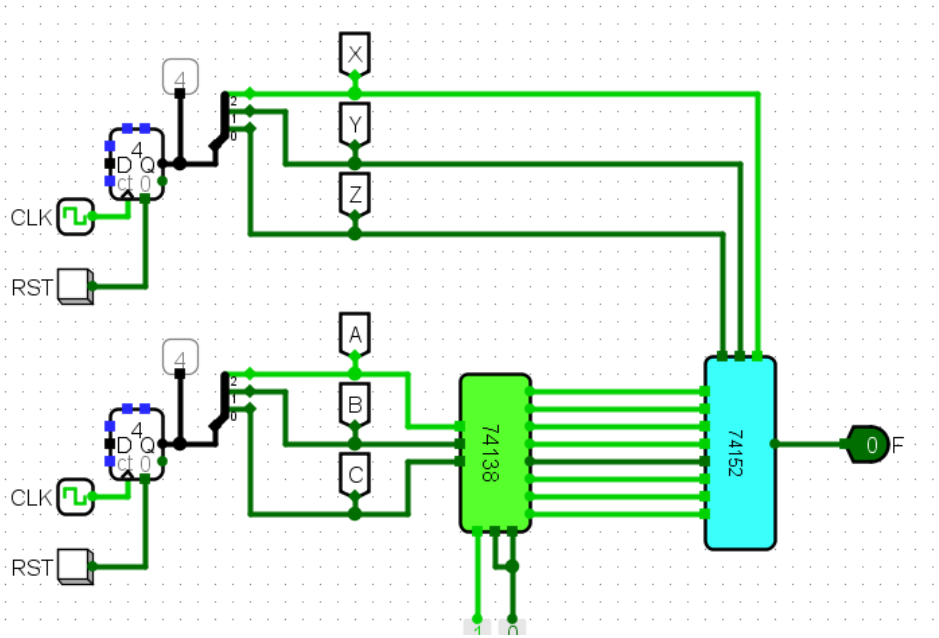


图 7.22　比较器的逻辑电路



（二）在Logisim上实现教材第7章的习题

- **1、（设计实验，课后完成）习题7.1的实现**
  - 请在Logisim上实现习题7.1的电路，并验证该电路的功能。

7.1　用 4 位二进制并行加法器设计一个实现 8421 码对 9 求补的逻辑电路。

所谓对9求补，就是9减这个数

例如，5对9求补=9-5=4；10对9求补=9-10=-1

- **2、（设计实验，课后完成）习题7.2的实现**
  - 请在Logisim上实现习题7.2的电路，并验证该电路的功能。

7.2　用两个4位二进制并行加法器实现2位十进制数8421码到二进制码的转换。

例如，78的8421码=0111 1000；78的二进制码=0100 1110

该电路就是能将“0111 1000”转换为“0100 1110”

- **3、（设计实验，课后完成）习题7.3的实现**
  - 请在Logisim上实现习题7.3的电路，并验证该电路的功能。

7.3　用 4 位二进制并行加法器设计一个用 8421 码表示的 1 位十进制加法器。

该电路的输入为：0000～1001（0～9），输出为00～18（0000 0000～0001 1000）

即输入为两个1位8421码（两个4位二进制数），输出为一个2位8421码（一个8位二进制数）

- 4、（设计实验，课后完成）习题7.4的实现
  - 请在Logisim上实现习题7.4的电路，并验证该电路的功能。

7.4 用一片 3-8 线译码器和必要的逻辑门实现下列逻辑函数表达式：

$$F_1 = \overline{A}\,\overline{C} + AB\overline{C}$$

$$F_2 = \overline{A} + B$$

$$F_3 = AB + \overline{A}\,\overline{B}$$

请参考例题7.6的验证实验

- 9、（验证实验）例题7.6的实现
  - 在Logisim上实现例题7.6的逻辑电路（图7.9）：F(A,B,C,D)= ∑m(2,4,6,8,10,12,14)。
  - 验证步骤：
    - 先按复位按钮RST，再按时钟按钮CLK多次，观看是否得到正确的结果（输入为2、4、6、8、10、12、14时，输出=1；其余情况，输出=0）？

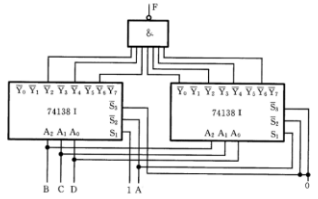
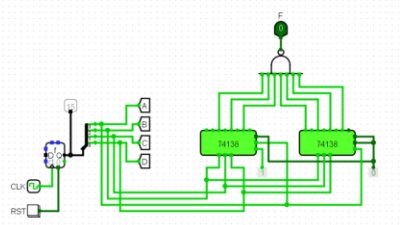


图 7.9 逻辑电路



- **5、（设计实验，课后完成）习题7.6的实现**
  - 请在Logisim上实现习题7.6的电路，并验证该电路的功能。

7.6　当优先编码器 74LS148 的$\overline{I}_s$ 接 0，输入$\overline{I}_7$ $\overline{I}_6$ $\overline{I}_5$ $\overline{I}_4$ $\overline{I}_3$ $\overline{I}_2$ $\overline{I}_1$ $\overline{I}_0$ = 11010001 时，输出状态为何值？

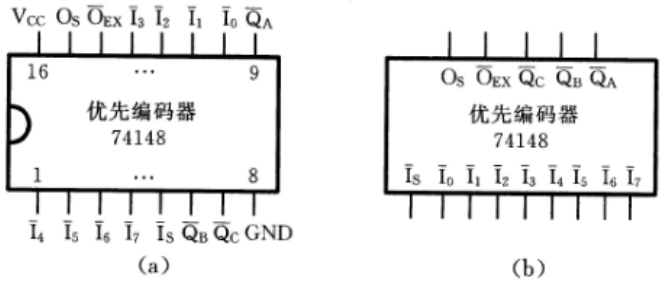


图 7.14　74148 优先编码器引脚排列和逻辑符号

表 7.6　74148 真值表

输入									输出				
$\overline{I}_S$	$\overline{I}_0$	$\overline{I}_1$	$\overline{I}_2$	$\overline{I}_3$	$\overline{I}_4$	$\overline{I}_5$	$\overline{I}_6$	$\overline{I}_7$	$\overline{Q}_C$	$\overline{Q}_B$	$\overline{Q}_A$	$\overline{Q}_{EX}$	O_S
1	d	d	d	d	d	d	d	d	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	d	d	d	0	0	0	0	0	1
0	d	d	d	d	d	d	0	1	0	0	1	0	1
0	d	d	d	d	d	0	1	1	0	1	0	0	1
0	d	d	d	d	0	1	1	1	0	1	1	0	1
0	d	d	d	0	1	1	1	1	1	0	0	0	1
0	d	d	0	1	1	1	1	1	1	0	1	0	1
0	d	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

- 6、（设计实验，课后完成）习题7.7的实现
  - 请在Logisim上实现习题7.7的电路，并验证该电路的功能。

7.7 试用 4 路数据选择器实现余 3 码到 8421 码的转换。

表 1.3 常用的 3 种 BCD 码

十进制字符	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

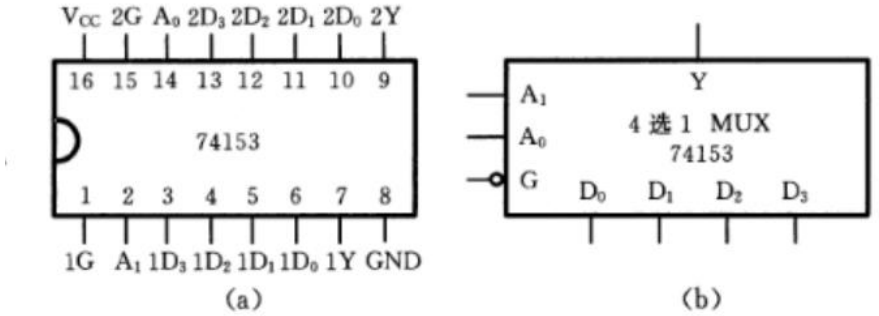


图 7.16 74153 引脚排列图和逻辑符号

74153为双4路选择器MUX芯片

请参考例题7.8、7.9、7.10的验证实验

- 7、（设计实验，课后完成）习题7.8的实现
  - 请在Logisim上实现习题7.8的电路，并验证该电路的功能。

7.8　当4路选择器的选择控制变量 A_1、A_0 接变量 A、B，数据输入端 D_0、D_1、D_2、D_3 依次接 $\overline{C}$、0、0、C 时，电路实现何功能？

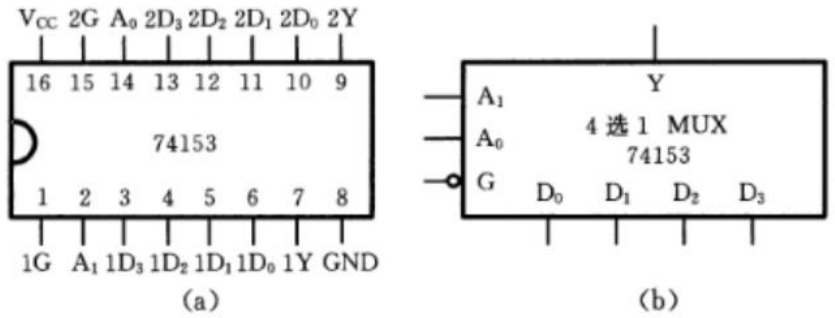


图 7.16　74153 引脚排列图和逻辑符号

74153为双4路选择器MUX芯片

- **8、（选择实验）习题7.5的实现**
  - **请在Logisim上实现习题7.5的电路，并验证该电路的功能。**

7.5　用一片 4-16 线译码器和适当的逻辑门设计一个 1 位十进制数 2421 码的奇偶位产生电路(假定采用奇检验)。

第二部分：在FPGA开发板上实现中规模组合逻辑电路

请将“第6次实验（发给学生）\EGO1”目录中的所有文件拷贝到“\DigitalLogic\EGO1”目录中

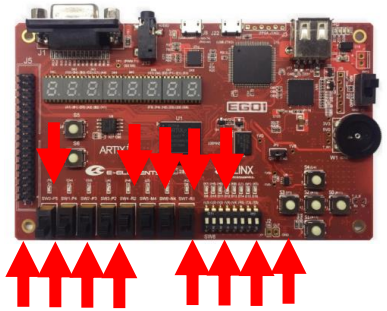
（一）在FPGA开发板上实现教材第7章的例题

- 1、（验证实验）4位二进制串行加法器

  - 采用结构化描述方式实现4位二进制串行加法器。该电路的输入为A（4位）和B（4位），输出为F（4位）和C4。假设C0=0。

  - 输入A为开发板上左边的4个拨动开关（左边为最高位），输入B为开发板上右边的4个拨动开关（左边为最高位），输出F为开发板上最右边的4个LED灯（左边为最高位），输出C4为开发板上最左边的1个LED灯。

  - 验证步骤：
    - 运行程序后，通过8个拨动开关设置不同的输入A和B，观看5个LED灯的显示情况（C4，F）是否正确？



EGO1开发板

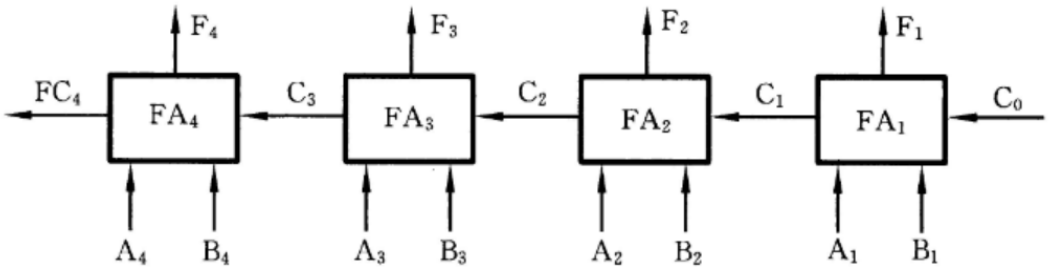


图 7.1　4 位串行进位二进制并行加法器的结构框图

```
//结构化描述方式（子模块采用行为描述方式）实现4位二进制串行加法器，输入A为开发板上的左边的4个拨动开关，输入B为开发板上的右边的4个拨动开关，输出C4为开发板上最左边的1个LED灯，输出F为开发板上最右边的4个LED

`timescale 1ns / 1ps

module full_adder(
    input a,
    input b,
    input cin,
    output f,
    output cout
);
    reg y1, y2;

    always @(*)                    //行为描述方式
    begin
        {y2,y1} <= a+b+cin;
    end

    assign f = y1;
    assign cout = y2;

endmodule

module serial_adder(
    input a3, a2, a1, a0, b3, b2, b1, b0, cin,
    output f3, f2, f1 ,f0, cout
);
    wire c1, c2, c3;

    full_adder U1(.a(a0),.b(b0),.cin(cin),.f(f0),.cout(c1));          //结构化描述方式
    full_adder U2(.a(a1),.b(b1),.cin(c1),.f(f1),.cout(c2));           //结构化描述方式
    full_adder U3(.a(a2),.b(b2),.cin(c2),.f(f2),.cout(c3));           //结构化描述方式
    full_adder U4(.a(a3),.b(b3),.cin(c3),.f(f3),.cout(cout));         //结构化描述方式

endmodule

module serial_adder_exe(
    input sw_pin[7:0],                                 //8个拨动开关
    output [15:0] led_pin                              //16个led灯
);

    serial_adder U(.a3(sw_pin[0]), .a2(sw_pin[1]), .a1(sw_pin[2]), .a0(sw_pin[3]), .b3(sw_pin[4]), .b2(sw_pin[5]), .b1(sw_pin[6]), .b0(sw_pin[7]), .cin(0), .f3(led_pin[4]), .f2(led_pin[5]), .f1(led_pin[6]), .f0(led_pin[7]), .cout(led_pin

endmodule
```

工程名：serial_adder_EGO1

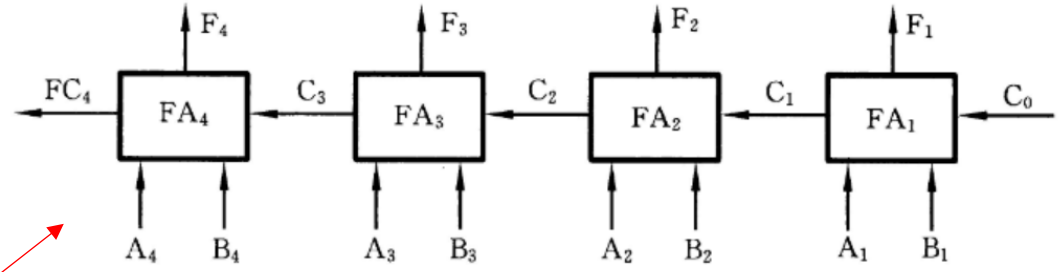


图 7.1　4 位串行进位二进制并行加法器的结构框图

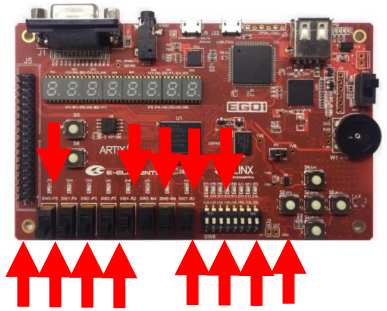
结构化描述方式

• 2、（验证实验）4位二进制并行加法器（超前进位）

- 采用行为描述方式实现4位二进制并行加法器（超前进位）。该电路的输入为A（4位）和B（4位），输出为F（4位）和C4。假设C0=0。

- 输入A为开发板上左边的4个拨动开关（左边为最高位），输入B为开发板上右边的4个拨动开关（左边为最高位），输出F为开发板上最右边的4个LED灯（左边为最高位），输出C4为开发板上最左边的1个LED灯。

- 验证步骤：
  - 运行程序后，通过8个拨动开关设置不同的输入A和B，观看5个LED灯的显示情况（C4，F）是否正确？



EGO1开发板

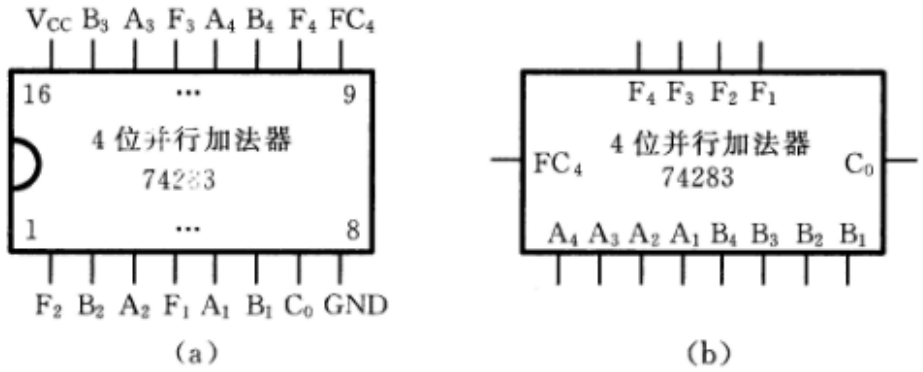


图 7.2　74283 的引脚排列图和逻辑符号

carry_lookahead_adder_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现4位二进制并行加法器（超前进位），输入A为开发板上的左边的4个拨动开关，输入B为开发板上的右边的4个拨动开关，输出C4为开发板上最左边的1个LED灯，输出F为开发板上最右边的4个LED灯。

`timescale 1ns / 1ps

module carry_lookahead_adder(
    input a3, a2, a1, a0, b3, b2, b1, b0, cin,
    output f3, f2, f1 ,f0, cout
);

    reg y3, y2, y1, y0, c4, c3, c2, c1;

    always @(*)                    //行为描述方式
    begin
        {c1,y0} <= a0+b0+cin;
        {c2,y1} <= a1+b1+c1;
        {c3,y2} <= a2+b2+c2;
        {c4,y3} <= a3+b3+c3;
    end

    assign f0 = y0;
    assign f1 = y1;
    assign f2 = y2;
    assign f3 = y3;

    assign cout = c4;

endmodule

module carry_lookahead_adder_exe(
    input sw_pin[7:0],                              //8个拨动开关
    output [15:0] led_pin                           //16个led灯
);

    carry_lookahead_adder U(.a3(sw_pin[0]), .a2(sw_pin[1]), .a1(sw_pin[2]), .a0(sw_pin[3]), .b3(sw_pin[4]), .b2(sw_pin[5]), .b1(sw_pin[6]), .b0(sw_pin[7]), .cin(0), .f3(led_pin[4]), .f2(led_pin[5]), .f1(led_pin[6]), .f0(led_pin[7]), .c

endmodule
```

工程名：carry_lookahead_adder_EGO1

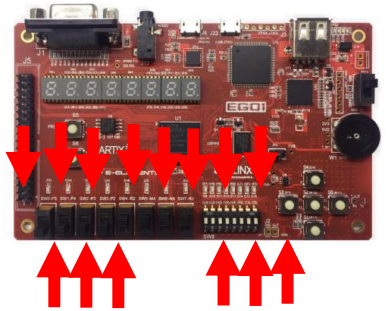
行为描述方式

• 3、（验证实验）3-8译码器

- 采用行为描述方式实现3-8译码器（74138芯片）。该电路的输入为（S_1、/S_2、/S_3）和（A_2、A_1、A_0），输出为（/Y_0、/Y_1、/Y_2、/Y_3、/Y_4、/Y_5、/Y_6、/Y_7）。

- 输入（S_1、/S_2、/S_3）为开发板上的最左边的3个拨动开关，输入（A_2、A_1、A_0）为开发板上的最右边的3个拨动开关，输出（/Y_0、/Y_1、/Y_2、/Y_3、/Y_4、/Y_5、/Y_6、/Y_7）为开发板上的8个LED灯。

- 验证步骤：
  - 运行程序后，通过6个拨动开关设置不同的输入（S_1、/S_2、/S_3；A_2、A_1、A_0），观看8个LED灯的显示情况（/Y_0、/Y_1、/Y_2、/Y_3、/Y_4、/Y_5、/Y_6、/Y_7）是否与表7.1相同?



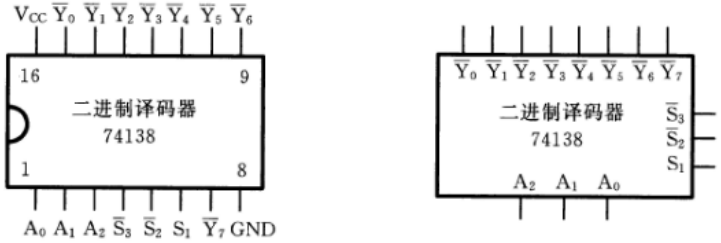


图 7.7　74138 译码器的引脚排列图和逻辑符号

表 7.1　74138 译码器真值表

输入					输出							
S_1	$\overline{S}_2+\overline{S}_3$	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	1	1	1	1	1	1	1	1
d	1	d	d	d	1	1	1	1	1	1	1	1

74138_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现74138（3-8译码器）芯片，输入（S1、/S2、/S3）为开发板上的最左边的3个

`timescale 1ns / 1ps

module ttl74138(
    input s1, s2, s3, a2, a1, a0,
    output y0, y1, y2, y3, y4, y5, y6, y7
);
    wire s23;
    reg f0, f1, f2, f3, f4, f5, f6, f7;

    assign s23 = s2 | s3;

    always @(*)                    //行为描述方式
    begin
        if(s1==1 & s23==0 & a2==0 & a1==0 & a0==0)
            begin f0 <= 0; f1 <= 1; f2 <= 1; f3 <= 1; f4 <= 1; f5 <= 1; f6 <= 1; f7 <= 1; end

        if(s1==1 & s23==0 & a2==0 & a1==0 & a0==1)
            begin f0 <= 1; f1 <= 0; f2 <= 1; f3 <= 1; f4 <= 1; f5 <= 1; f6 <= 1; f7 <= 1; end

        if(s1==1 & s23==0 & a2==0 & a1==1 & a0==0)
            begin f0 <= 1; f1 <= 1; f2 <= 0; f3 <= 1; f4 <= 1; f5 <= 1; f6 <= 1; f7 <= 1; end

        if(s1==1 & s23==0 & a2==0 & a1==1 & a0==1)
            begin f0 <= 1; f1 <= 1; f2 <= 1; f3 <= 0; f4 <= 1; f5 <= 1; f6 <= 1; f7 <= 1; end

        if(s1==1 & s23==0 & a2==1 & a1==0 & a0==0)
            begin f0 <= 1; f1 <= 1; f2 <= 1; f3 <= 1; f4 <= 0; f5 <= 1; f6 <= 1; f7 <= 1; end

        if(s1==1 & s23==0 & a2==1 & a1==0 & a0==1)
            begin f0 <= 1; f1 <= 1; f2 <= 1; f3 <= 1; f4 <= 1; f5 <= 0; f6 <= 1; f7 <= 1; end

        if(s1==1 & s23==0 & a2==1 & a1==1 & a0==0)
            begin f0 <= 1; f1 <= 1; f2 <= 1; f3 <= 1; f4 <= 1; f5 <= 1; f6 <= 0; f7 <= 1; end

        if(s1==1 & s23==0 & a2==1 & a1==1 & a0==1)
            begin f0 <= 1; f1 <= 1; f2 <= 1; f3 <= 1; f4 <= 1; f5 <= 1; f6 <= 1; f7 <= 0; end

        if(s1==0)
            begin f0 <= 1; f1 <= 1; f2 <= 1; f3 <= 1; f4 <= 1; f5 <= 1; f6 <= 1; f7 <= 1; end

        if(s23==1)
            begin f0 <= 1; f1 <= 1; f2 <= 1; f3 <= 1; f4 <= 1; f5 <= 1; f6 <= 1; f7 <= 1; end
    end
```

工程名：74138_EGO1

行为描述方式

表 7.1　74138 译码器真值表

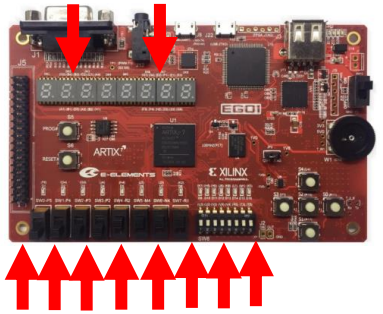
输入					输出							
S_1	$\bar{S}_2+\bar{S}_3$	A_2	A_1	A_0	$\bar{Y}_0$	$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$	$\bar{Y}_4$	$\bar{Y}_5$	$\bar{Y}_6$	$\bar{Y}_7$
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	1	1	1	1	1	1	1	1
d	1	d	d	d	1	1	1	1	1	1	1	1

• 4、（验证实验）七段显示译码器

- 采用行为描述方式实现七段显示译码器。该电路的输入为A（4位二进制数）和B（4位二进制数），输出为在数码管上显示A和B对应的十六进制值。

- 输入A为开发板上左边的4个拨动开关，输入B为开发板上右边的4个拨动开关，输出为开发板上的8个数码管（左边4个对应A的十六进制值，右边4个对应B的十六进制值）。

- 验证步骤：
  - 运行程序后，通过8个拨动开关设置不同的输入A和B，观看8（6）个数码管显示的情况是否正确？



EGO1开发板

显示字符	共阴极段选码	共阳极段选码	显示字符	共阴极段选码	共阳极段选码
0	3FH	C0H	C	39H	C6H
1	06H	F9H	d	5EH	A1H
2	5BH	A4H	E	79H	86H
3	4FH	B0H	F	71H	84H
4	66H	99H			
5	6DH	92H			
6	7DH	82H			
7	07H	F8H			
8	7FH	80H			
9	6FH	90H	“灭”	00H	FFH
A	77H	88H			⋮
b	7CH	83H			

- EGO1开发板的数码管的控制信号：
  - seg_data_0_pin[7:0]：左边4个数码管的段信号（共阴极）
  - seg_data_1_pin[7:0]：右边4个数码管的段信号（共阴极）
  - seg_cs_pin[7:0]：8个数码管的位选择信号（高电平有效，最高位对应最右边的数码管，最低位对应最左边的数码管）

- 例如，要在最左边的数码管上显示“9”，则：
  - seg_data_0_pin=8’h6f
  - seg_cs_pin=8’b00000001

- 例如，要在最右边的数码管上显示“1”，则：
  - seg_data_0_pin=8’h06
  - seg_cs_pin=8’b10000000

- 例如，要在左边的4个数码管上都显示“9”，则：
  - seg_data_0_pin=8’h6f
  - seg_cs_pin=8’b00001111

- 例如，要在右边的4个数码管上都显示“1”，则：
  - seg_data_0_pin=8’h06
  - seg_cs_pin=8’b11110000

显示字符	共阴极段选码	共阳极段选码	显示字符	共阴极段选码	共阳极段选码
0	3FH	C0H	C	39H	C6H
1	06H	F9H	d	5EH	A1H
2	5BH	A4H	E	79H	86H
3	4FH	B0H	F	71H	84H
4	66H	99H			
5	6DH	92H			
6	7DH	82H			
7	07H	F8H			
8	7FH	80H			
9	6FH	90H	“灭”	00H	FFH
A	77H	88H			⋮
b	7CH	83H			

segment_display_decoder_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现七段显示译码器。输入A为开发板上左边的4个拨动开关，输入B为开发板上右边的4个拨动

`timescale 1ns / 1ps

module segment_display_decoder(
    input [7:0] sw_pin,                                     //8个拨动开关
    output [7:0] seg_data_0_pin, seg_data_1_pin, seg_cs_pin  //8个数码管
);

    reg [7:0] seg0, seg1;

    always @(*)                 //行为描述方式
    begin
        case({sw_pin[0],sw_pin[1],sw_pin[2],sw_pin[3]})
                0:      seg0 <= 8'h3f;                  // 左边的4个数码管显示“0”
                1:      seg0 <= 8'h06;                  // 左边的4个数码管显示“1”
                2:      seg0 <= 8'h5b;                  // 左边的4个数码管显示“2”
                3:      seg0 <= 8'h4f;                  // 左边的4个数码管显示“3”
                4:      seg0 <= 8'h66;                  // 左边的4个数码管显示“4”
                5:      seg0 <= 8'h6d;                  // 左边的4个数码管显示“5”
                6:      seg0 <= 8'h7d;                  // 左边的4个数码管显示“6”
                7:      seg0 <= 8'h07;                  // 左边的4个数码管显示“7”
                8:      seg0 <= 8'h7f;                  // 左边的4个数码管显示“8”
                9:      seg0 <= 8'h6f;                  // 左边的4个数码管显示“9”
                10:     seg0 <= 8'h77;                  // 左边的4个数码管显示“A”
                11:     seg0 <= 8'h7c;                  // 左边的4个数码管显示“b”
                12:     seg0 <= 8'h39;                  // 左边的4个数码管显示“c”
                13:     seg0 <= 8'h5e;                  // 左边的4个数码管显示“d”
                14:     seg0 <= 8'h79;                  // 左边的4个数码管显示“E”
                15:     seg0 <= 8'h71;                  // 左边的4个数码管显示“F”
                default:  seg0 <= 8'h00;                // 左边的4个数码管全灭
        endcase

        case({sw_pin[4],sw_pin[5],sw_pin[6],sw_pin[7]})
                0:      seg1 <= 8'h3f;                  // 右边的4个数码管显示“0”
                1:      seg1 <= 8'h06;                  // 右边的4个数码管显示“1”
                2:      seg1 <= 8'h5b;                  // 右边的4个数码管显示“2”
                3:      seg1 <= 8'h4f;                  // 右边的4个数码管显示“3”
                4:      seg1 <= 8'h66;                  // 右边的4个数码管显示“4”
                5:      seg1 <= 8'h6d;                  // 右边的4个数码管显示“5”
                6:      seg1 <= 8'h7d;                  // 右边的4个数码管显示“6”
                7:      seg1 <= 8'h07;                  // 右边的4个数码管显示“7”
                8:      seg1 <= 8'h7f;                  // 右边的4个数码管显示“8”
                9:      seg1 <= 8'h6f;                  // 右边的4个数码管显示“9”
                10:     seg1 <= 8'h77;                  // 右边的4个数码管显示“A”
                11:     seg1 <= 8'h7c;                  // 右边的4个数码管显示“b”
                12:     seg1 <= 8'h39;                  // 右边的4个数码管显示“c”
                13:     seg1 <= 8'h5e;                  // 右边的4个数码管显示“d”
```

工程名：segment_display_decoder_EGO1

显示字符	共阴极段选码	共阳极段选码	显示字符	共阴极段选码	共阳极段选码
0	3FH	C0H	C	39H	C6H
1	06H	F9H	d	5EH	A1H
2	5BH	A4H	E	79H	86H
3	4FH	B0H	F	71H	84H
4	66H	99H			
5	6DH	92H			
6	7DH	82H			
7	07H	F8H			
8	7FH	80H			
9	6FH	90H	“灭”	00H	FFH
A	77H	88H			⋮
b	7CH	83H			

行为描述方式

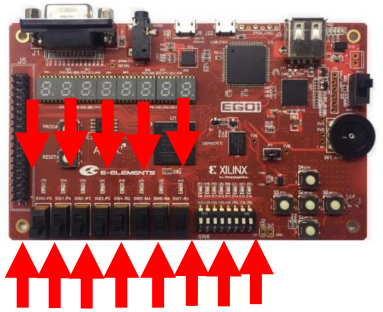
```
    assign seg_data_0_pin = seg0;          //左边4个数码管的8个段
    assign seg_data_1_pin = seg1;          //右边4个数码管的8个段
    assign seg_cs_pin = 8'hff;             //8个数码管的8个位

endmodule
```

• 5、（验证实验）优先编码器

- 采用行为描述方式实现8-3线优先编码器（74148芯片）。该电路的输入为（/I_0、/I_1、/I_2、/I_3、/I_4、/I_5、/I_6、/I_7），输出为（/Q_C、/Q_B、/Q_A、/Q_{EX}、O_S），假设/I_S=0。

- 输入（/I_0、/I_1、/I_2、/I_3、/I_4、/I_5、/I_6、/I_7）为开发板上的8个拨动开关，输出（/Q_C、/Q_B、/Q_A、/Q_{EX}、O_S）为开发板上最左边的5个LED灯。

- 验证步骤：
  - 运行程序后，通过8个拨动开关设置不同的输入（/I_0、/I_1、/I_2、/I_3、/I_4、/I_5、/I_6、/I_7），观看5个LED灯的显示情况（/Q_C、/Q_B、/Q_A、/Q_{EX}、O_S）是否与表7.6相同？



EGO1开发板

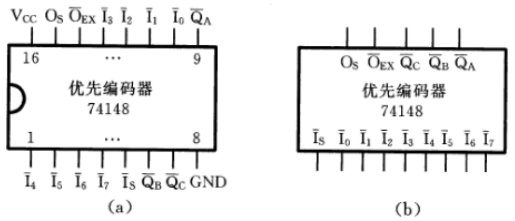


图 7.14　74148 优先编码器引脚排列和逻辑符号

表 7.6　74148 真值表

输入									输出				
$\overline{I}_S$	$\overline{I}_0$	$\overline{I}_1$	$\overline{I}_2$	$\overline{I}_3$	$\overline{I}_4$	$\overline{I}_5$	$\overline{I}_6$	$\overline{I}_7$	$\overline{Q}_C$	$\overline{Q}_B$	$\overline{Q}_A$	$\overline{Q}_{EX}$	O_S
1	d	d	d	d	d	d	d	d	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	d	d	d	0	0	0	0	0	1
0	d	d	d	d	d	d	0	1	0	0	1	0	1
0	d	d	d	d	d	0	1	1	0	1	0	0	1
0	d	d	d	d	0	1	1	1	0	1	1	0	1
0	d	d	d	0	1	1	1	1	1	0	0	0	1
0	d	d	0	1	1	1	1	1	1	0	1	0	1
0	d	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

74148_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现74148（8-3线优先编码器）芯片，输入（/I0、/I1、/I2、/I3、/I4、/I5、/I

`timescale 1ns / 1ps

module ttl74148(
    input is, i0, i1, i2, i3, i4, i5, i6, i7,
    output qc, qb, qa, qex, os
);
    reg fqc, fqb, fqa, fqex, fos;

    always @(*)                    //行为描述方式
    begin
        if(is==1)
            begin fqc <= 1; fqb <= 1; fqa <= 1; fqex <= 1; fos <= 1; end

        if(is==0 & i0==1 & i1==1 & i2==1 & i3==1 & i4==1 & i5==1 & i6==1 & i7==1)
            begin fqc <= 1; fqb <= 1; fqa <= 1; fqex <= 1; fos <= 0; end

        if(is==0 &  i7==0)
            begin fqc <= 0; fqb <= 0; fqa <= 0; fqex <= 0; fos <= 1; end

        if(is==0 &  i6==0 & i7==1)
            begin fqc <= 0; fqb <= 0; fqa <= 1; fqex <= 0; fos <= 1; end

        if(is==0 & i5==0 & i6==1 & i7==1)
            begin fqc <= 0; fqb <= 1; fqa <= 0; fqex <= 0; fos <= 1; end

        if(is==0 & i4==0 & i5==1 & i6==1 & i7==1)
            begin fqc <= 0; fqb <= 1; fqa <= 1; fqex <= 0; fos <= 1; end

        if(is==0 &  i3==0 & i4==1 & i5==1 & i6==1 & i7==1)
            begin fqc <= 1; fqb <= 0; fqa <= 0; fqex <= 0; fos <= 1; end

        if(is==0 & i2==0 & i3==1 & i4==1 & i5==1 & i6==1 & i7==1)
            begin fqc <= 1; fqb <= 0; fqa <= 1; fqex <= 0; fos <= 1; end

        if(is==0 & i1==0 & i2==1 & i3==1 & i4==1 & i5==1 & i6==1 & i7==1)
            begin fqc <= 1; fqb <= 1; fqa <= 0; fqex <= 0; fos <= 1; end

        if(is==0 & i0==0 & i1==1 & i2==1 & i3==1 & i4==1 & i5==1 & i6==1 & i7==1)
            begin fqc <= 1; fqb <= 1; fqa <= 1; fqex <= 0; fos <= 1; end
    end
```

工程名：74148_EGO1

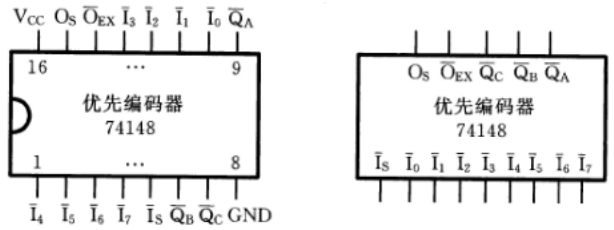


图 7.14　74148 优先编码器引脚排列和逻辑符号

表 7.6　74148 真值表

输入									输出				
$\overline{I}_S$	$\overline{I}_0$	$\overline{I}_1$	$\overline{I}_2$	$\overline{I}_3$	$\overline{I}_4$	$\overline{I}_5$	$\overline{I}_6$	$\overline{I}_7$	$\overline{Q}_C$	$\overline{Q}_B$	$\overline{Q}_A$	$\overline{Q}_{EX}$	O_S
1	d	d	d	d	d	d	d	d	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	d	d	d	0	0	0	0	0	1
0	d	d	d	d	d	d	0	1	0	0	1	0	1
0	d	d	d	d	d	0	1	1	0	1	0	0	1
0	d	d	d	d	0	1	1	1	0	1	1	0	1
0	d	d	d	0	1	1	1	1	1	0	0	0	1
0	d	d	0	1	1	1	1	1	1	0	1	0	1
0	d	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

行为描述方式

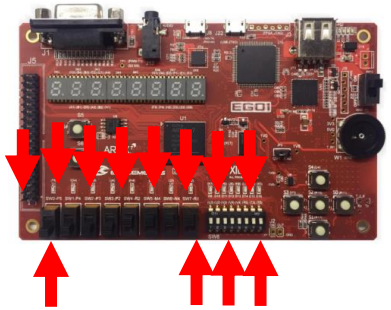
```
    assign qc = fqc;
    assign qb = fqb;
    assign qa = fqa;
    assign qex = fqex;
    assign os = fos;

endmodule

module ttl74148_exe(
    input sw_pin[7:0],                          //8个拨动开关
    output [15:0] led_pin                       //16个led灯
);

    ttl74148 U(.is(0), .i0(sw_pin[0]), .i1(sw_pin[1]), .i2(sw_pin[2]), .i3(sw_pin[3]), .i4(sw_pin[4]), .i5(sw_pin[5]), .i6(sw_pin[6]), .i7(sw_pin[7]), .qc(led_pin[0]), .qb(led_pin[1]), .qa(led_pin[2]), .qex(led_pin[3]), .os(led_pin[4]));

endmodule
```

• 6、（验证实验）8路多路分配器（DEMUX）

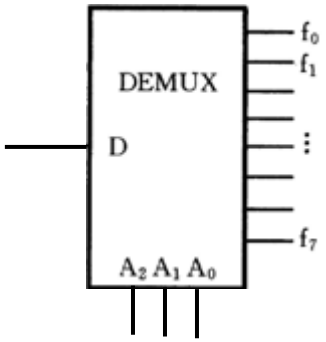
- 采用行为描述方式实现8路多路分配器（DEMUX）。该电路的输入为（D）和（A_2、A_1、A_0），输出为（f_0、f_1、f_2、f_3、f_4、f_5、f_6、f_7）。

- 输入（D）为开发板上的最左边的1个拨动开关，输入（A_2、A_1、A_0）为开发板上的最右边的3个拨动开关，输出（f_0、f_1、f_2、f_3、f_4、f_5、f_6、f_7）为开发板上的8个LED灯。

- 验证步骤：
  - 运行程序后，通过4个拨动开关设置不同的输入（D；A_2、A_1、A_0），观看8个LED灯的显示情况（f_0、f_1、f_2、f_3、f_4、f_5、f_6、f_7）是否与下表相同?



EGO1开发板



输入			输出							
A_2	A_2	A_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	D	0	0	0	0	0	0	0
0	0	1	0	D	0	0	0	0	0	0
0	1	0	0	0	D	0	0	0	0	0
0	1	1	0	0	0	D	0	0	0	0
1	0	0	0	0	0	0	D	0	0	0
1	0	1	0	0	0	0	0	D	0	0
1	1	0	0	0	0	0	0	0	D	0
1	1	1	0	0	0	0	0	0	0	D

*demux8_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现8路多路分配器（DEMUX），输入（D）为开发板上的最左边的1个拨动开关，输

`timescale 1ns / 1ps

module demux8(
    input d, a2, a1, a0,
    output f0, f1, f2, f3, f4, f5, f6, f7
);
    reg y0, y1, y2, y3, y4, y5, y6, y7;

    always @(*)                    //行为描述方式
    begin
        if(a2==0 & a1==0 & a0==0)
            begin y0 <= d; y1 <= 0; y2 <= 0; y3 <= 0; y4 <= 0; y5 <= 0; y6 <= 0; y7 <= 0; end

        if(a2==0 & a1==0 & a0==1)
            begin y0 <= 0; y1 <= d; y2 <= 0; y3 <= 0; y4 <= 0; y5 <= 0; y6 <= 0; y7 <= 0; end

        if(a2==0 & a1==1 & a0==0)
            begin y0 <= 0; y1 <= 0; y2 <= d; y3 <= 0; y4 <= 0; y5 <= 0; y6 <= 0; y7 <= 0; end

        if(a2==0 & a1==1 & a0==1)
            begin y0 <= 0; y1 <= 0; y2 <= 0; y3 <= d; y4 <= 0; y5 <= 0; y6 <= 0; y7 <= 0; end

        if(a2==1 & a1==0 & a0==0)
            begin y0 <= 0; y1 <= 0; y2 <= 0; y3 <= 0; y4 <= d; y5 <= 0; y6 <= 0; y7 <= 0; end

        if(a2==1 & a1==0 & a0==1)
            begin y0 <= 0; y1 <= 0; y2 <= 0; y3 <= 0; y4 <= 0; y5 <= d; y6 <= 0; y7 <= 0; end

        if(a2==1 & a1==1 & a0==0)
            begin y0 <= 0; y1 <= 0; y2 <= 0; y3 <= 0; y4 <= 0; y5 <= 0; y6 <= d; y7 <= 0; end

        if(a2==1 & a1==1 & a0==1)
            begin y0 <= 0; y1 <= 0; y2 <= 0; y3 <= 0; y4 <= 0; y5 <= 0; y6 <= 0; y7 <= d; end
    end
```

工程名：demux8_EGO1

输入			输出							
A_2	A_2	A_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	D	0	0	0	0	0	0	0
0	0	1	0	D	0	0	0	0	0	0
0	1	0	0	0	D	0	0	0	0	0
0	1	1	0	0	0	D	0	0	0	0
1	0	0	0	0	0	0	D	0	0	0
1	0	1	0	0	0	0	0	D	0	0
1	1	0	0	0	0	0	0	0	D	0
1	1	1	0	0	0	0	0	0	0	D

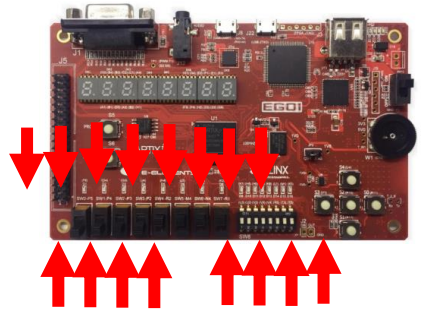
行为描述方式

• 7、（设计实验，课后完成）4位二进制乘法器

- 采用结构化描述方式实现4位二进制乘法器（图7.6）。该电路的输入为A（4位）和B（4位），输出为F（8位）。工程文件命名：multiplier_EGO1。

- 输入A为开发板上左边的4个拨动开关（左边为最高位），输入B为开发板上右边的4个拨动开关（左边为最高位），输出F为开发板上的8个LED灯（左边为最高位）。

- 验证方法：
  - 运行程序后，通过8个拨动开关设置不同的输入A和B，观看8个LED灯的显示情况（F）是否正确？



EGO1开发板

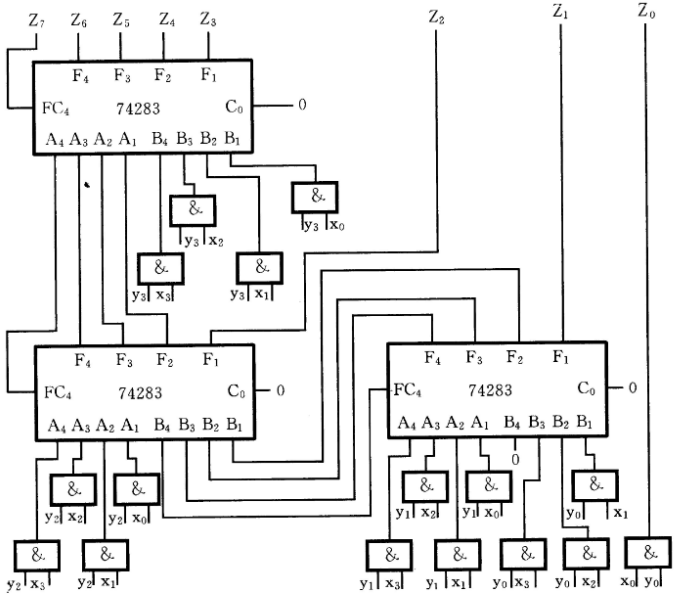
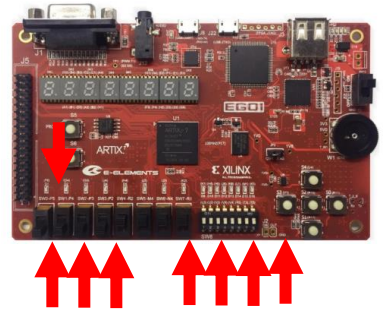


图 7.6　逻辑电路

- 8、（设计实验，课后完成）4选1多路选择器（MUX）
  - 采用行为描述方式实现4选1多路选择器（MUX，74153芯片的一半）。该电路的输入为（G、A_1、A_0）和（D_0、D_1、D_2、D_3），输出为Y。工程文件命名：74153_EGO1。
  - 输入（G、A_1、A_0）为开发板上的最左边的3个拨动开关，输入（D_0、D_1、D_2、D_3）为开发板上的最右边的4个拨动开关，输出（Y）为开发板上最左边的1个LED灯。
  - 验证方法：
    - 运行程序后，通过7个拨动开关设置不同的输入（G、A_1、A_0；D_0、D_1、D_2、D_3），观看1个LED灯的显示情况（Y）是否与表7.7相同？



EGO1开发板

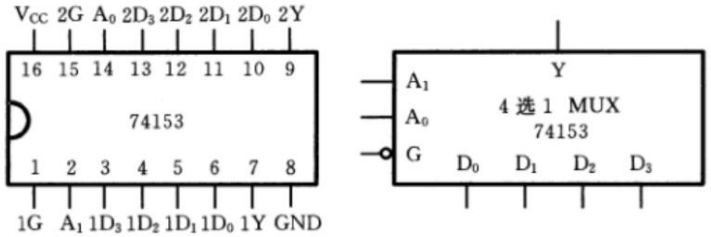


图 7.16　74153 引脚排列图和逻辑符号

表 7.7　4 路 MUX 74153 的功能表

使能输入	选择输入		数据输入				输　出
G	A_1	A_0	D_0	D_1	D_2	D_3	Y
1	d	d	d	d	d	d	0
0	0	0	D_0	d	d	d	D_0
0	0	1	d	D_1	d	d	D_1
0	1	0	d	d	D_2	d	D_2
0	1	1	d	d	d	D_3	D_3

（二）在FPGA开发板上实现教材第7章的习题

- **1、（选做实验）习题7.1的实现**
  - **请在FPGA开发板上实现习题7.1的电路，并验证该电路的功能。**

7.1　用 4 位二进制并行加法器设计一个实现 8421 码对 9 求补的逻辑电路。

所谓对9求补，就是9减这个数

例如，5对9求补=9-5=4；10对9求补=9-10=-1

- **2、（选做实验）习题7.2的实现**
  - **请在FPGA开发板上实现习题7.2的电路，并验证该电路的功能。**

7.2　用两个4位二进制并行加法器实现2位十进制数8421码到二进制码的转换。

例如，78的8421码=0111 1000；78的二进制码=0100 1110

该电路就是能将“0111 1000”转换为“0100 1110”

- **3、（选做实验）习题7.3的实现**
  - 请在FPGA开发板上实现习题7.3的电路，并验证该电路的功能。

7.3　用 4 位二进制并行加法器设计一个用 8421 码表示的 1 位十进制加法器。

该电路的输入为：0000～1001（0～9），输出为00～18（0000 0000～0001 1000）

即输入为两个1位8421码（两个4位二进制数），输出为一个2位8421码（一个8位二进制数）

- 4、（选做实验）习题7.4的实现
  - 请在FPGA开发板上实现习题7.4的电路，并验证该电路的功能。

7.4 用一片 3-8 线译码器和必要的逻辑门实现下列逻辑函数表达式：

$$F_1 = \overline{A}\,\overline{C} + AB\overline{C}$$

$$F_2 = \overline{A} + B$$

$$F_3 = AB + \overline{A}\,\overline{B}$$

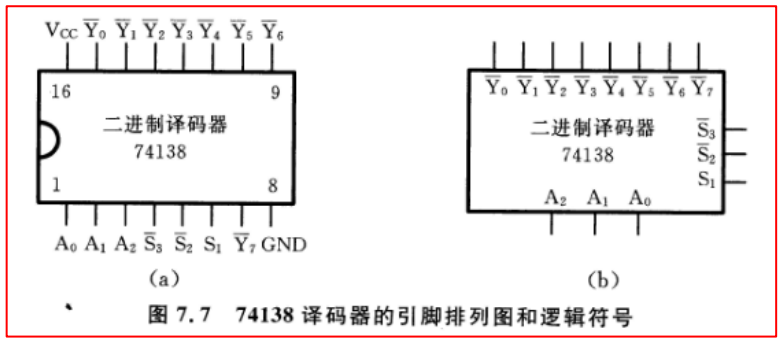


图 7.7 74138 译码器的引脚排列图和逻辑符号

表 7.1 74138 译码器真值表

输入					输出							
S_1	$\overline{S}_2+\overline{S}_3$	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0
0	d	d	d	d	1	1	1	1	1	1	1	1
d	1	d	d	d	1	1	1	1	1	1	1	1

3-8线译码器

- **5、（选做实验）习题7.5的实现**
  - **请在FPGA开发板上实现习题7.5的电路，并验证该电路的功能。**

7.5　用一片 4-16 线译码器和适当的逻辑门设计一个 1 位十进制数 2421 码的奇偶位产生电路(假定采用奇检验)。

- 6、（选做实验）习题7.7的实现
  - 请在Logisim上实现习题7.7的电路，并验证该电路的功能。

7.7　试用 4 路数据选择器实现余 3 码到 8421 码的转换。

表 1.3　常用的 3 种 BCD 码

十进制字符	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

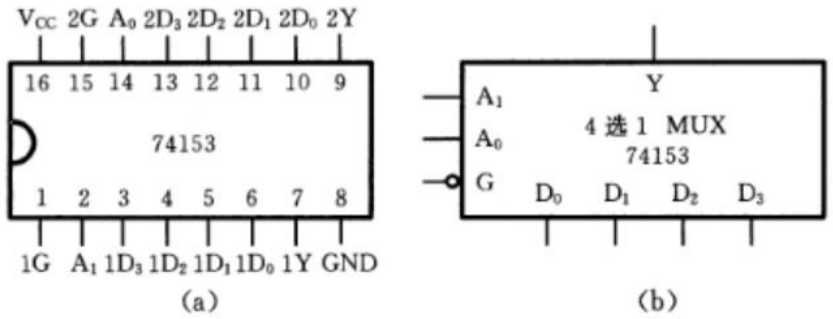


图 7.16　74153 引脚排列图和逻辑符号

74153为双4路选择器MUX芯片

实验要求

- 1、在Logisim上完成验证实验，将实验过程中的主要结果通过截屏（或拍照）的方式，拷贝到实验报告中，并给予适当的文字说明。
- 2、在FPGA开发板上完成验证实验，将实验过程中的主要结果通过截屏（或拍照）的方式，拷贝到实验报告中，并给予适当的文字说明。
- 3、在Logisim上完成设计实验，设计文件命名为：第7章习题电路的实现.circ。
- 4、在FPGA开发板上完成设计实验，工程文件、设计文件、约束文件请严格按照规定的要求命名。
- 5、实验报告命名为：学号+姓名+第6次实验报告.docx。
- 6、将设计文件、实验报告打包压缩成1个压缩文件，命名为：学号+姓名+第6次实验.zip（或.rar），并上传到FTP上，上传截止日期：2024年12月1日晚上24点。
- 7、鼓励有兴趣的同学完成选做实验。

Thanks