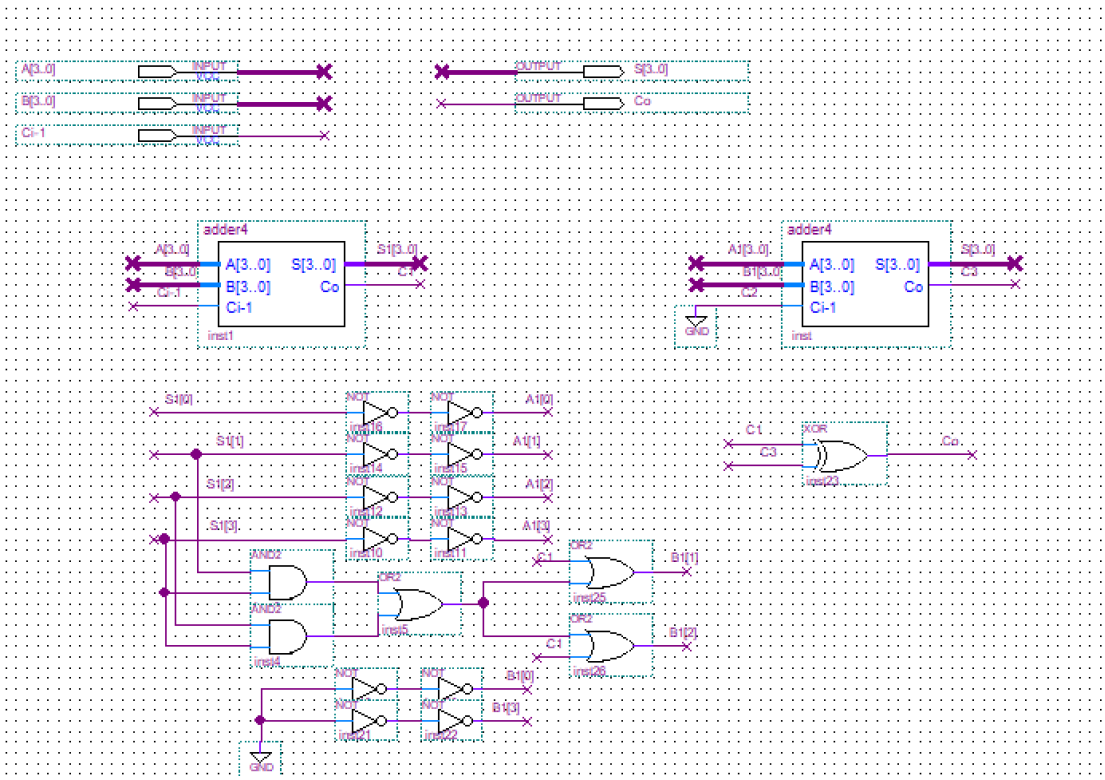
1， BCD 加减器器设计

先设计 BCD 加法器，利用原本的四位全加器实现，即当若原本输出的结果大于 10 时，利用加法器自动加六，改进原本的全加器如下，可以先用一个全加器进行二进制加减，再用一次全加器转化为 BCD 码，其中 D0 的逻辑表达式为 D10 = CO + B3B2 + B3B1

由此可以设计出 BCD 加法器

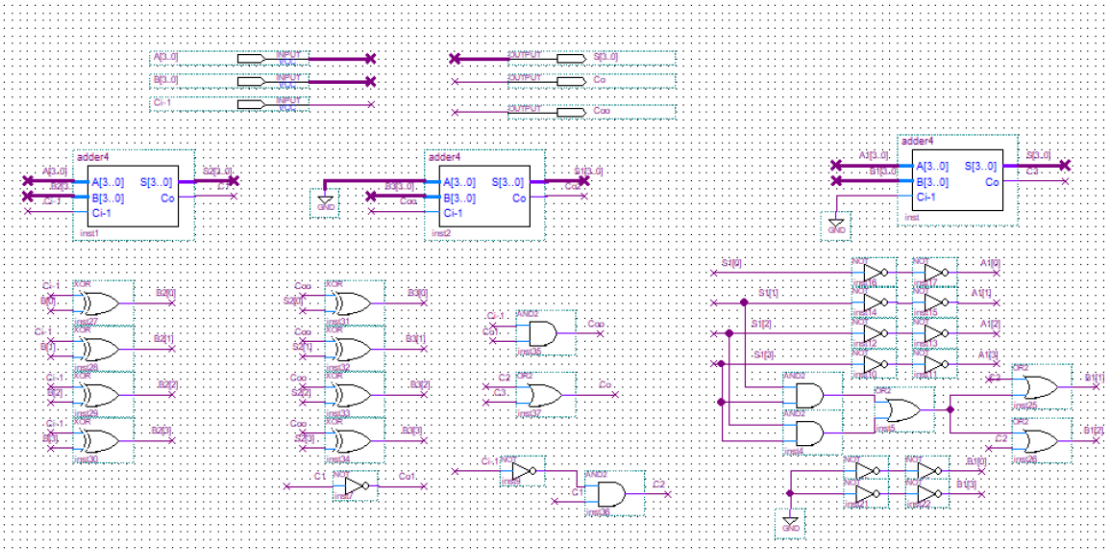


我觉得以上设计中最关键的一点就是要针对进位的情况，如果相加后出现进位就一定要额外增加 6，否则对于 9+9 的情况会出现错误，具体就是表达式为 Co=C1 或 C3，表格如下

C1	C3	Co
0	0	0
0	1	1
1	0	1
1	1	1

对以上方案进行修改，增加减法功能

我们可以首先通过最开始的逻辑判断，如果是加那就直接相加，如果是减就取补码再相加，然后我们可以在中间增加一个加法器，用于针对减法的补码，将其重新转化成原码，而如果是正数那就不进行变化，最后和原来一样再利用一次加法器实现对 BCD 码的转化



对于以上设计，我觉得最关键的地方在于符号位和进位的处理，对于符号位实际上是由最开始的逻辑变量和第一次加法的进位共同决定的，如下表格

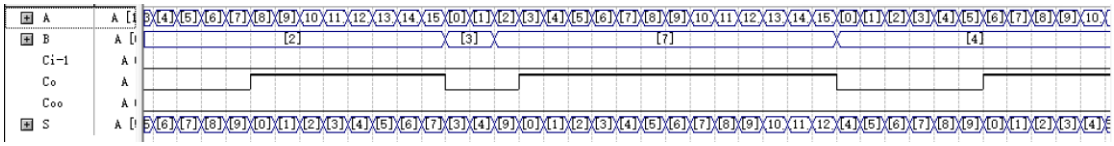
逻辑变量 Ci-1	进位 C1	C1 取反	Coo 正数与否（0 正）
0	1	0	0
0	0	1	0
1	1	0	0
1	0	1	1

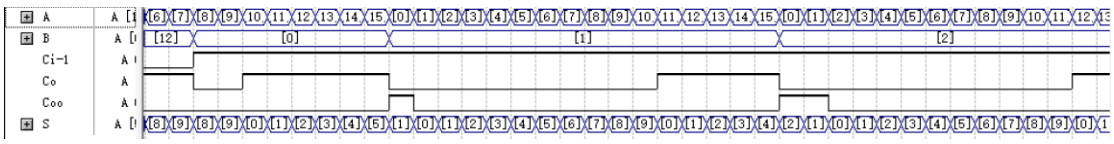
可见如果对 C1 取反符号位被简化成与运算 Coo=（C1 非）与 Ci-1

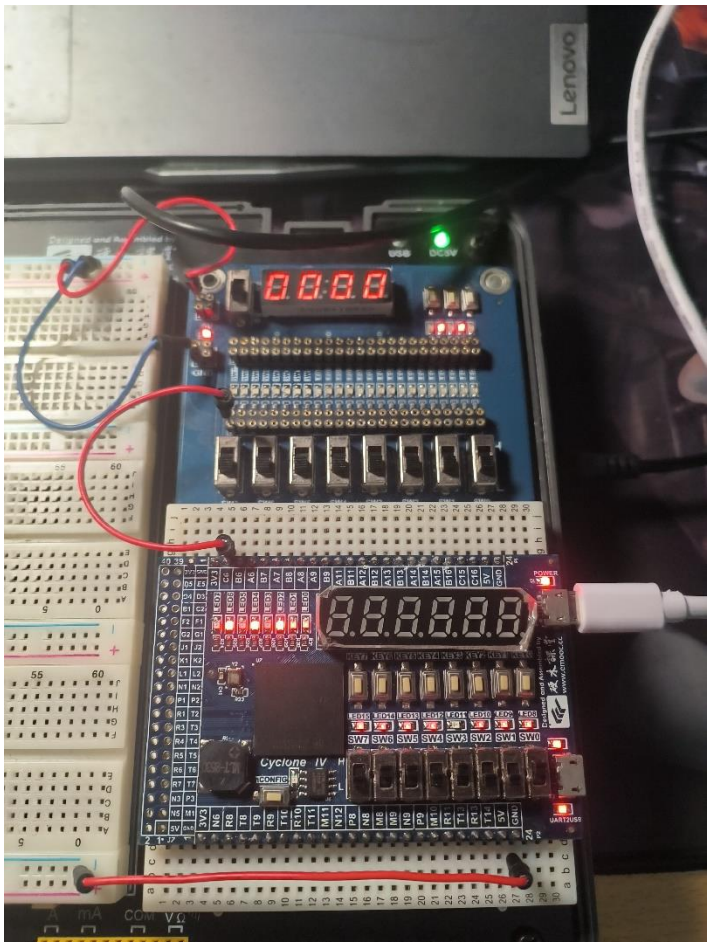
而进位有两个来源，一个是第一个的加法器运算，一个是 BCD 导致的进位，BCD 进位源自于最后一个加法器的进位 C3，而第一给加法器进位来自于 C1，显然要考虑减法时的情况，表格如下：

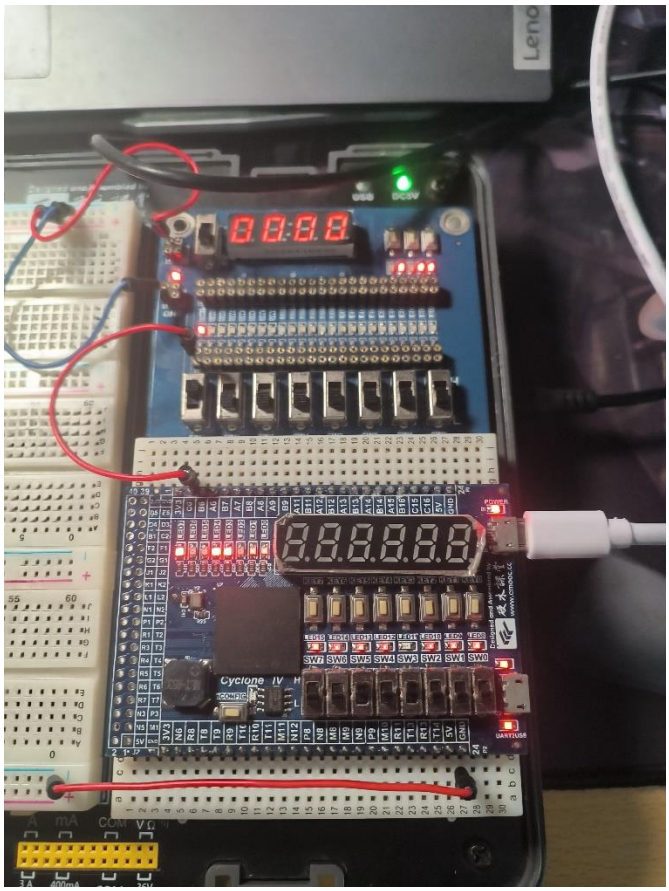
Ci-1	C1	C3	Co
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

所以可见 Co=((Ci-1 非)与 C1)或 C3






Lenovo
Cyclone IV
SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0


Cyclone IV
SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0