## 数字逻辑实验报告

姓名：詹冲 学号：2023211616

### 一、8421码和格雷码的转换

1.实验内容：

利用VHDL语言设计一个4位8421码到4位格雷码的编码器。使用波形图仿真验证其功能。

2.代码：

library ieee;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity test\_1 is

port(a:in std\_logic\_vector(3 downto 0);

b:out std\_logic\_vector(3 downto 0));

end entity test\_1;

architecture change of test\_1 is

begin

b(3)<=a(3);

b(2)<=a(3) OR a(2);

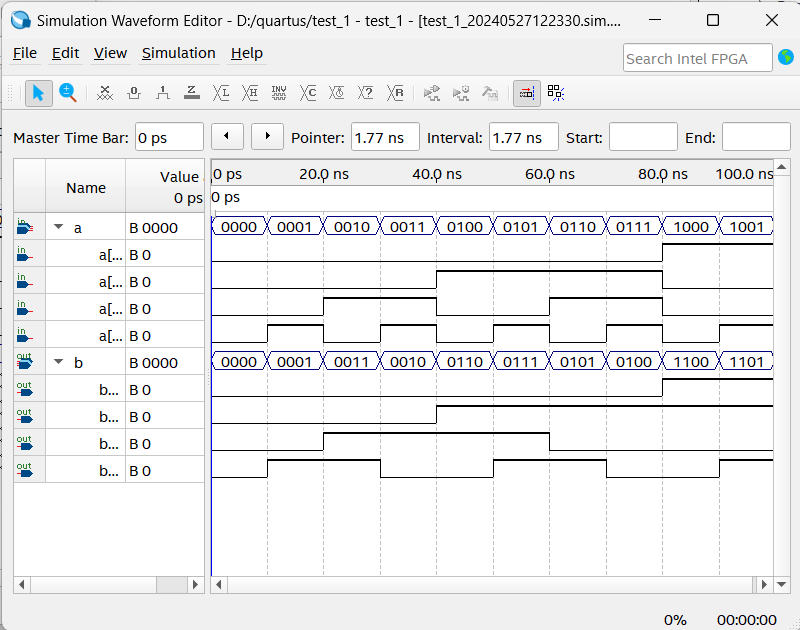
b(1)<=a(2) XOR a(1);

b(0)<=a(1) XOR a(0);

end architecture;

解释：根据真值表和卡诺图的化简可以得到b3=a3,b2=a2+a3,b1=a2xora1,b0=a1xora0。直接在代码中体现即可。

3.波形图：



4.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test\_1的实体（entity），该实体具有一个输入端口a和一个输出端口b。输入端口a是一个4位宽的std\_logic\_vector，表示它可以接收4个逻辑值（0或1）。输出端口b也是一个4位宽的std\_logic\_vector，用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它将输入端口a的值进行一些逻辑操作，并将结果赋值给输出端口b。以下是每个位的操作：

①b(3) <= a(3)：将输入端口a的第3位（从右向左数，从0开始计数）的值直接赋给输出端口b的第3位。

②b(2) <= a(3) OR a(2)：将输入端口a的第3位和第2位进行逻辑或操作，然后将结果赋给输出端口b的第2位。

③b(1) <= a(2) XOR a(1)：将输入端口a的第2位和第1位进行异或操作，然后将结果赋给输出端口b的第1位。

④b(0) <= a(1) XOR a(0)：将输入端口a的第1位和第0位进行异或操作，然后将结果赋给输出端口b的第0位。

总之，这段代码实现了一个简单的逻辑操作电路，根据输入端口a的值计算输出端口b的值。

### 二、数值比较器

1.实验介绍：

数值比较器是对两个位数相同的二进制数进行比较并判定其大小关系的算术运算电路。2.实验内容：

使用if语句编写对两个4位二进制数进行比较的VHDL程序，其中A和B分别是参与比较的两个4位二进制数，YA、YB、YC是用来分别表示A>B、A<B、A=B的3个输出端。使用波形图仿真验证其功能。

3.代码：

library ieee;

use ieee.std\_logic\_1164.all;

entity test\_2 is

port(a,b:in std\_logic\_vector(3 downto 0);

ya,yb,yc:out std\_logic);

end entity test\_2;

architecture contrast of test\_2 is

begin

process (a,b)

begin

if(a>b) then

ya<='1';

yb<='0';

yc<='0';

elsif(a<b) then

ya<='0';

yb<='1';

yc<='0';

else

ya<='0';

yb<='0';

yc<='1';

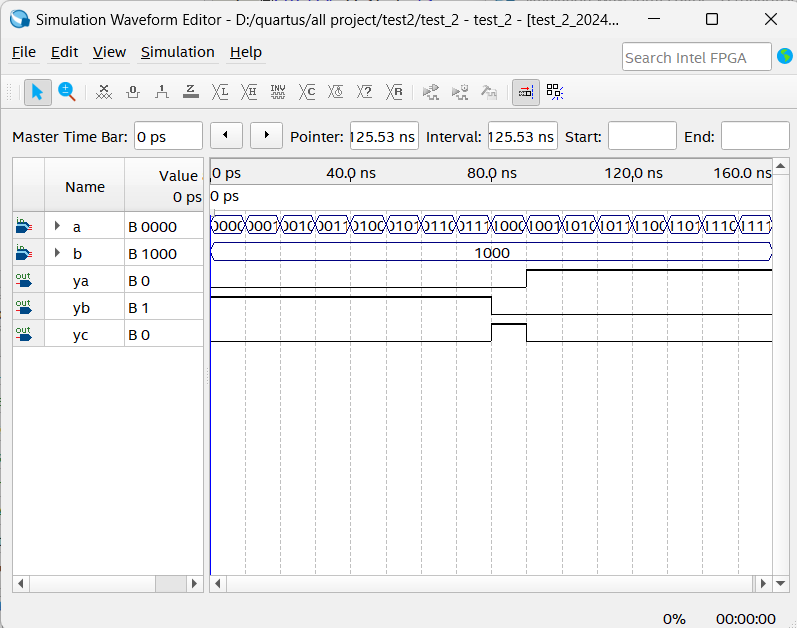
end if;

end process;

end architecture;

解释：创建两个四位二进制a,b，创建三个标识符ya,yb,yc分别代表a>b,a=b,a<b,在代码中体现即可。

4.波形图



5.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test\_2的实体（entity），该实体具有两个输入端口a和b，以及三个输出端口ya、yb和yc。输入端口a和b都是4位宽的std\_logic\_vector，表示它们可以接收4个逻辑值（0或1）。输出端口ya、yb和yc都是1位宽的std\_logic，用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它使用一个process语句来比较输入端口a和b的值，并根据比较结果设置输出端口ya、yb和yc的值。以下是比较规则：

①如果a大于b，则将ya设置为'1'，yb设置为'0'，yc设置为'0'。

②如果a小于b，则将ya设置为'0'，yb设置为'1'，yc设置为'0'。

③如果a等于b，则将ya设置为'0'，yb设置为'0'，yc设置为'1'。

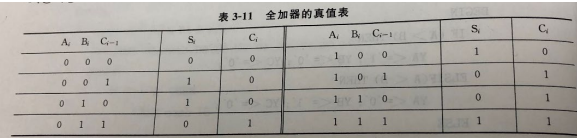
总之，这段代码实现了一个简单的比较器电路，根据输入端口a和b的值计算输出端口ya、yb和yc的值。

### 三、全加器

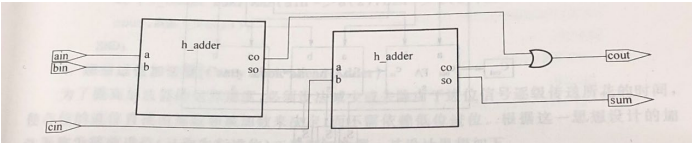
1.实验介绍：

把加数、被加数和低位进位逻辑三者加起来的电路称为全加器。其逻辑表达式为Si=Ai⊕Bi⊕Ci-1，Ci=(Ai⊕Bi)Ci-1+AiBi。

其真值表如下表所示。



全加器可以由两个半加器和一个或门构成，其电路图如下图所示。



2.实验内容：

根据全加器的电路图和真值表，设计一个VHDL程序实现全加器。使用波形图仿真验证其功能。

3.代码：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_signed.all;

entity test\_3 is

port(a,b,c0:in std\_logic;

c1,s:out std\_logic);

end entity test\_3;

architecture total\_adder of test\_3 is

signal tmp:std\_logic;

begin

s<=a xor b xor c0;

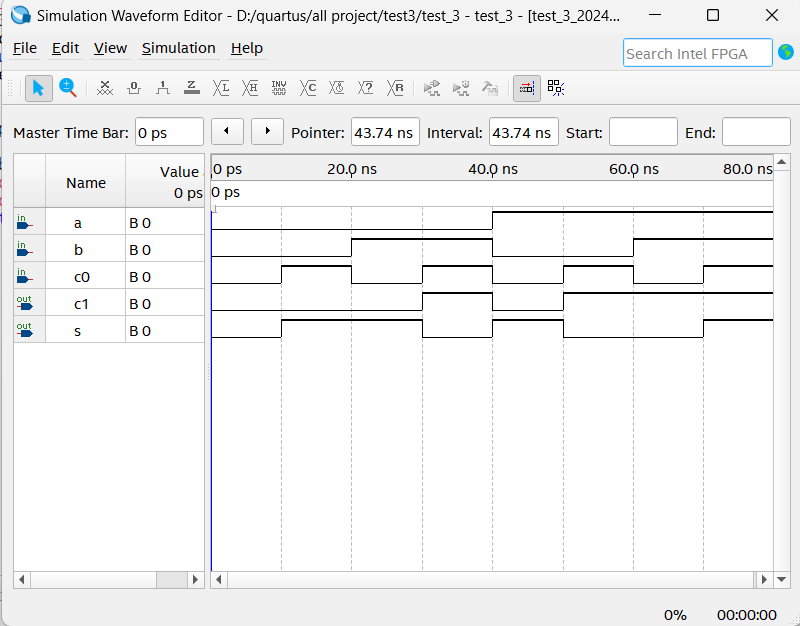
tmp<=(a xor b)and c0;

c1<= tmp or (a and b);

end architecture;

解释：根据全加器的真值表，得到s=a xor b xor c0,c1=((a xor b)c0) + ab,直接在代码中体现即可。

4.波形图：



5.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test\_3的实体（entity），该实体具有三个输入端口a、b和c0，以及两个输出端口c1和s。输入端口a、b和c0都是1位宽的std\_logic，表示它们可以接收一个逻辑值（0或1）。输出端口c1和s也是1位宽的std\_logic，用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它使用了一些逻辑运算符来计算输出端口c1和s的值。以下是计算规则：

①s <= a XOR b XOR c0：将输入端口a、b和c0进行异或操作，然后将结果赋给输出端口s。

②tmp <= (a XOR b) AND c0：将输入端口a和b进行异或操作，然后将结果与输入端口c0进行与操作，最后将结果赋给信号tmp。

③c1 <= tmp OR (a AND b)：将信号tmp与输入端口a和b进行与操作的结果进行或操作，然后将结果赋给输出端口c1。

总之，这段代码实现了一个简单的全加器电路，根据输入端口a、b和c0的值计算输出端口c1和s的值。

### 四、3线-8线译码器

1.实验背景:

译码器是一个多输入、多输出的组合逻辑电路。它的作用是把给定的代码进行“翻译”，变成相应的状态，使输出通道中相应的一路有信号输出。译码器在数字系统中有广泛的用途，不仅用于代码的转换、终端的数字显示，还用于数据分配、存储器寻址和组合控制信号等，不同的功能可选用不同种类的译码器。 译码器可分为通用译码器和数码显示译码器两大类，前者又分为变量译码器和代码变换3译码器。

2.设计任务:

描述一个3线-8线译码器，使能端为G1、G2A、G2B，地址选择端为A[2..0]，输出端为总线Y。

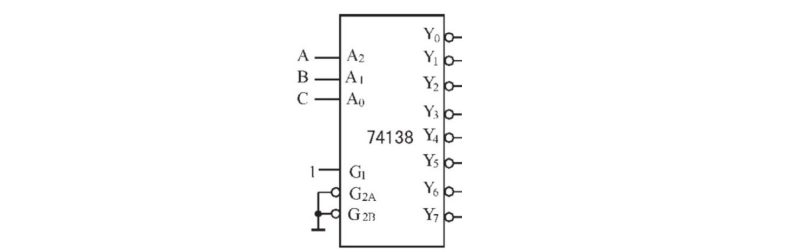


图1.3线-8线译码器74138的逻辑框图

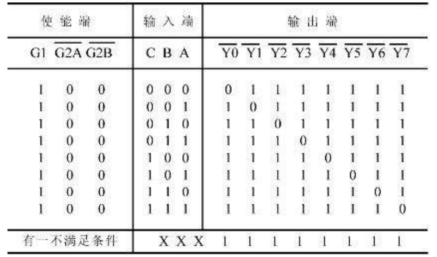


图2. 74138 的真值表

3.代码：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_signed.all;

entity test\_4 is

port(G2A,G2B,G1:in std\_logic;

a:in std\_logic\_vector(2 downto 0);

y:out std\_logic\_vector(7 downto 0));

end entity test\_4;

architecture L3\_8 of test\_4 is

begin

process(G1,G2A,G2B,a)

begin

if(G2A='1' and G2B='1' and G1='1') then

case a is

when "000"=> y<="00000001";

when "001"=> y<="00000010";

when "010"=> y<="00000100";

when "011"=> y<="00001000";

when "100"=> y<="00010000";

when "101"=> y<="00100000";

when "110"=> y<="01000000";

when "111"=> y<="10000000";

end case;

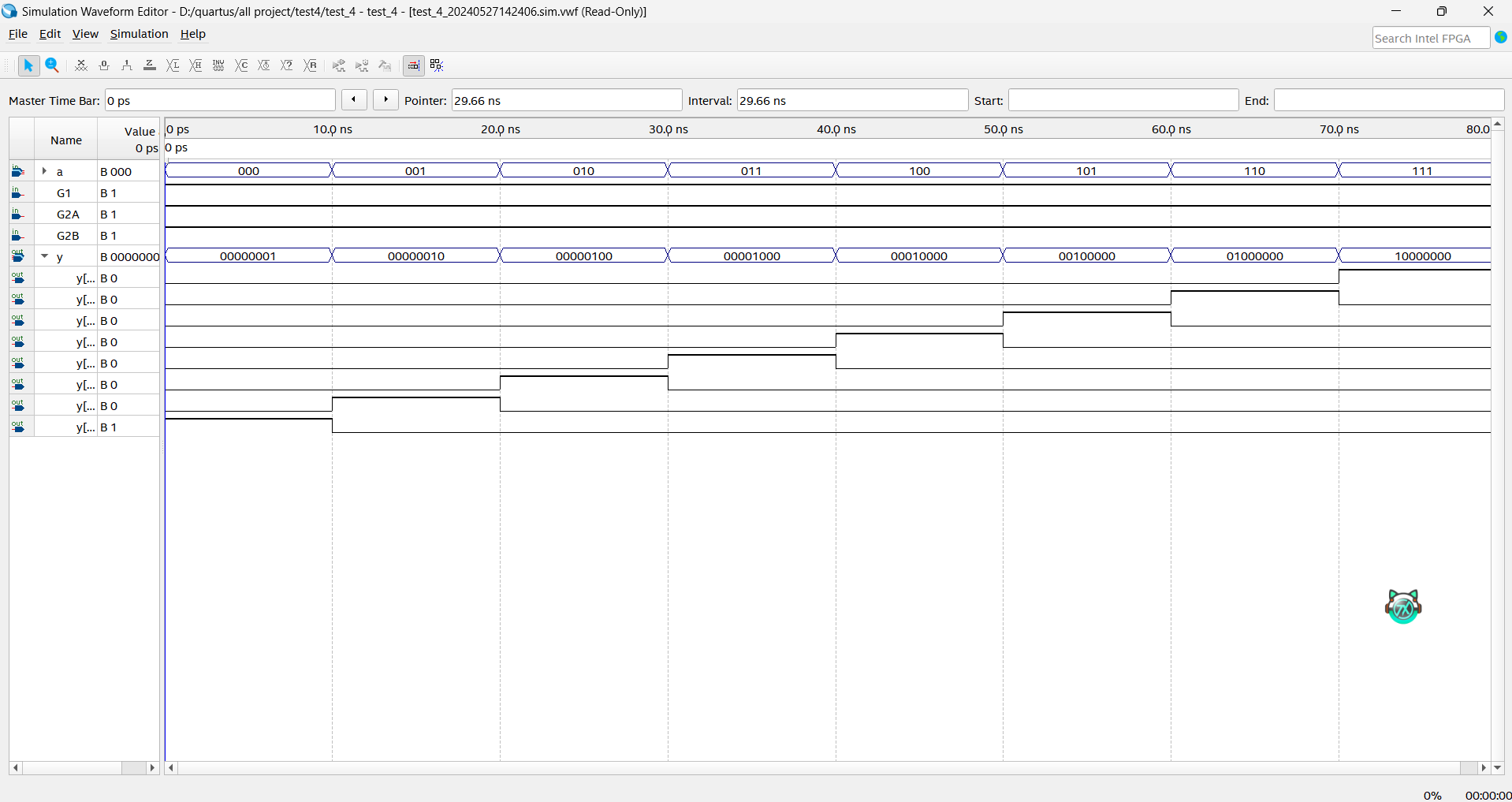
end if;

end process;

end architecture;

解释：根据真值表对应到代码中即可。

4.波形图：



5.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test\_4的实体（entity），该实体具有三个输入端口G2A、G2B和G1，以及一个输入端口a和一个输出端口y。输入端口G2A、G2B和G1都是1位宽的std\_logic，表示它们可以接收一个逻辑值（0或1）。输入端口a是一个3位宽的std\_logic\_vector，表示它可以接收三个逻辑值（0或1）。输出端口y是一个8位宽的std\_logic\_vector，用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它使用了一个process语句来根据输入端口G2A、G\*和G1的值以及输入端口a的值来计算输出端口y的值。以下是计算规则：

①如果G2A、G2B和G1的值都为'1'，则根据输入端口a的值进行case语句的判断。

②在case语句中，根据输入端口a的值（"000"、"001"、"010"、"011"、"100"、"101"、"110"、"111"），将对应的二进制数赋给输出端口y。例如，如果输入端口a的值为"000"，则输出端口y的值为"00000001"。

总之，这段代码实现了一个简单的多路选择器电路，根据输入端口G2A、G\*和G1的值以及输入端口a的值计算输出端口y的值。

### 五、表决器

1.实验内容：

用VHDL语言设计实现一个4人表决器，多数人赞成决议表示通过，否则决议不通过。 使用波形图仿真验证其功能。

2.代码：

library ieee;

use ieee.std\_logic\_1164.all;

entity test\_5 is

port(a,b,c,d:in std\_logic;

f:out std\_logic);

end entity test\_5;

architecture votor of test\_5 is

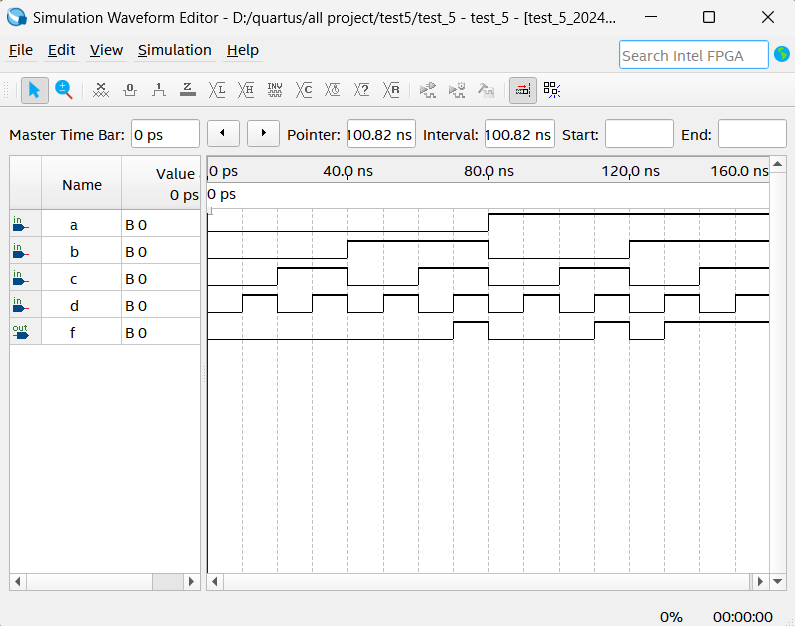
begin

f<=(a and b and c)or(a and b and d)or(a and c and d)or(b and c and d);

end architecture;

解释：根据功能需求，有三人或以上同意即可通过，所以f=abc+abd+acd+bcd,在代码中体现即可。

3.波形图：



4.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test\_5的实体（entity），该实体具有四个输入端口a、b、c和d，以及一个输出端口f。输入端口a、b、c和d都是1位宽的std\_logic，表示它们可以接收一个逻辑值（0或1）。输出端口f也是1位宽的std\_logic，用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它将输入端口a、b、c和d进行逻辑运算，然后将结果赋给输出端口f。以下是计算规则：

f <= (a AND b AND c) OR (a AND b AND d) OR (a AND c AND d) OR (b AND c AND d)

这个表达式的意思是，如果a、b、c中任意两个为1，或者a、b、d中任意两个为1，或者a、c、d中任意两个为1，或者b、c、d中任意两个为1，那么输出端口f的值为1，否则为0。

总之，这段代码实现了一个简单的四输入与门电路，根据输入端口a、b、c和d的值计算输出端口f的值。

### 六、序列检测器

1.实验内容：

设计一个序列检测器检测序列1110010。使用波形图进行仿真（至少要有一个检测成功 的波形）。

2.设计思路：

每输入一个序列1110010则会在输出端输出一个1，其余时间为0。首先写出状态转移图，再利用case语句，根据状态转移图写出状态的转移及输出。 检测序列为七位，所以可以设状态机状态数为8个； 输出信号由当前状态和输入条件决定，设计为mealy型状态机或moore型状态机，采用异步复位

3.代码：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity test\_6 is

port(clock,datain,reset:in std\_logic;

dataout:out std\_logic);

end test\_6;

architecture sequence\_detector of test\_6 is

type state\_type is (s1,s2,s3,s4,s5,s6,s7,s8);

signal state:state\_type;

begin

process(clock,reset)

begin

if reset='1' then state<=s1;

elsif clock'event and clock='1' then

case state is

when s1 => if datain='1' then

state<=s2;

end if;

when s2 => if datain='1' then

state<=s3;

end if;

when s3 => if datain='1' then

state<=s4;

end if;

when s4 => if datain='0' then

state<=s5;

end if;

when s5 => if datain='1' then

state<=s6;

end if;

when s6 => if datain='0' then

state<=s7;

end if;

when s7 => if datain='0' then

state<=s8;

end if;

when s8 => if datain='1' then

state<=s1;

end if;

end case;

end if;

end process;

process(state)

begin

case state is

when s8=>dataout<='1';

when others=>dataout<='0';

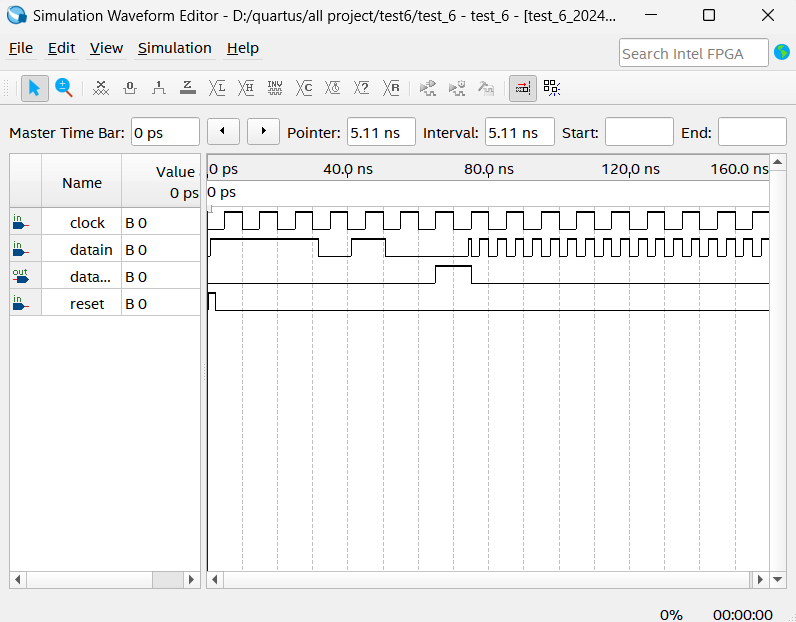
end case;

end process;

end sequence\_detector;

解释：根据需要检测的序列，在代码中体现输入和状态转移的关系即可。

4.波形图：



5.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test\_6的实体（entity），该实体具有三个输入端口clock、datain和reset，以及一个输出端口dataout。输入端口clock、datain和reset都是1位宽的std\_logic，表示它们可以接收一个逻辑值（0或1）。输出端口dataout也是1位宽的std\_logic，用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它使用了两个process语句来处理输入端口clock、datain和reset的变化，并根据这些变化更新状态变量state和输出端口dataout的值。

第一个process语句处理输入端口clock和reset的变化。当reset为'1'时，将状态变量state设置为s1；当clock发生上升沿（从0变为1）时，根据当前状态变量state的值进行状态转移。状态转移规则如下：

当state为s1时，如果datain为'1'，则将state设置为s2；

当state为s2时，如果datain为'1'，则将state设置为s3；

当state为s3时，如果datain为'1'，则将state设置为s4；

当state为s4时，如果datain为'0'，则将state设置为s5；

当state为s5时，如果datain为'1'，则将state设置为s6；

当state为s6时，如果datain为'0'，则将state设置为s7；

当state为s7时，如果datain为'0'，则将state设置为s8；

当state为s8时，如果datain为'1'，则将state设置为s1。

第二个process语句处理状态变量state的变化。根据当前状态变量state的值，设置输出端口dataout的值。规则如下：

当state为s8时，将dataout设置为'1'；

其他情况下，将dataout设置为'0'。

总之，这段代码实现了一个简单的序列检测器电路，根据输入端口clock、datain和reset的值计算输出端口dataout的值。

### 七、计数器

1.实验介绍

计数器是通过电路的状态反映驶入脉冲数目的电路。计数器是应用非常广泛的时序电路， 按照技术的特点分为二进制计数器、十进制计数器、环形计数器、扭环形计数器等。二进制 计数器又分为加计数器、减计数器等。

2.实验内容：

用VHDL语言实现计数状态连续的模值为2N的计数器。这是一个同步清零的4为二进制加计数器，其计数的状态是从“0000~1111”进行变化。整个的计数周期是 16 个时钟周期， 即24个时钟周期。凡是这种技术周期为2N且对计数状态无特殊要求的计数器，可以通过直接定义N位的计数信号和端口，对信号进行加或减操作，而不必进行计数状态的判断和控制。使用波形图进行仿真。

3.代码：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity test7 is

port (

clock: in std\_logic;

q: out std\_logic\_vector(3 downto 0);

c: out std\_logic

);

end test7;

architecture cnt of test7 is

begin

process(clock)

variable qi: std\_logic\_vector(3 downto 0) := (others => '0');

begin

if clock'event and clock = '0' then

if qi < "1111" then

qi := qi + 1;

else

qi := (others => '0');

end if;

if qi = "1111" then

c <= '1';

else

c <= '0';

end if;

end if;

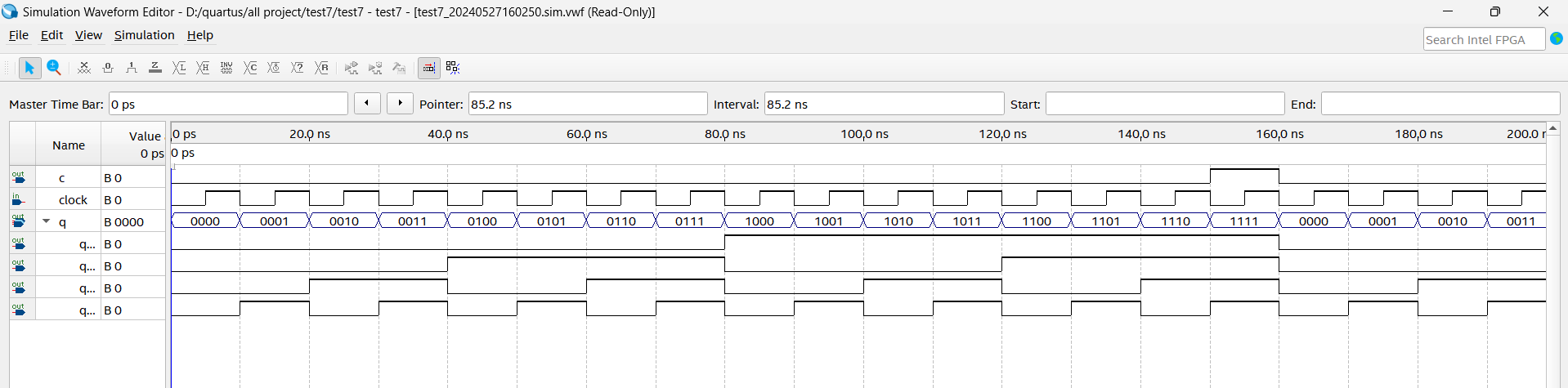
q <= qi;

end process;

end architecture;

解释：定义一个四位二进制数，每输入一个1，该二进制数就不断加1，当该二进制数为1111时，输出一个进位为1，该二进制数变为0。

4.波形图：



5.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test7的实体（entity），该实体具有一个输入端口clock，以及两个输出端口q和c。输入端口clock是1位宽的std\_logic，表示它可以接收一个逻辑值（0或1）。输出端口q是一个4位宽的std\_logic\_vector，用于输出结果；输出端口c是1位宽的std\_logic，也用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它使用了一个process语句来处理输入端口clock的变化，并根据这些变化更新变量qi、输出端口q和c的值。

当clock发生下降沿（从1变为0）时，执行以下操作：

①如果qi小于"1111"（即十进制的15），则将qi加1；否则，将qi设置为(others => '0')，即将qi的所有位都设置为0。

②如果qi等于"1111"，则将输出端口c设置为'1'；否则，将输出端口c设置为'0'。

将qi的值赋给输出端口q。

总之，这段代码实现了一个简单的计数器电路，根据输入端口clock的值计算输出端口q和c的值。当计数器的值达到最大值（即"1111"）时，输出端口c会被设置为'1'。

### 八、8位寄存器74374

1.实验介绍：

在数字系统中，寄存器可用来存储一组二进制代码，而触发器具有记忆功能，所以可以 用触发器构成寄存器。本实验要求同学们完成8位寄存器74374的VHDL描述。74374的逻辑框图如下图所示，功能表如下表所示。逻辑框图中 D 为寄存器的 8 位数据输入，Q 位寄 存器的8位数据输出端，CLK 为时钟信号，OE 为控制信号。从功能表可以看出 OE 为低电平时，在时钟上升沿输入端信号从输出端输出，其他时刻输出保持；而OE为高电平时，输出一直保持为高阻。

2.实验内容：

根据 74374 的逻辑框图和真值表，用 VHDL 语言实现 74374 的功能。并使用波形图进 行仿真。

3.代码：

library ieee;

use ieee.std\_logic\_1164.all;

entity test\_8 is

port(

clock, OE: in std\_logic;

D: in std\_logic\_vector(7 downto 0);

Q: out std\_logic\_vector(7 downto 0)

);

end entity test\_8;

architecture latch\_8 of test\_8 is

begin

process(clock)

begin

if OE = '0' then

if clock'event and clock = '1' then

Q <= D;

end if;

else

Q <= (others => 'Z');

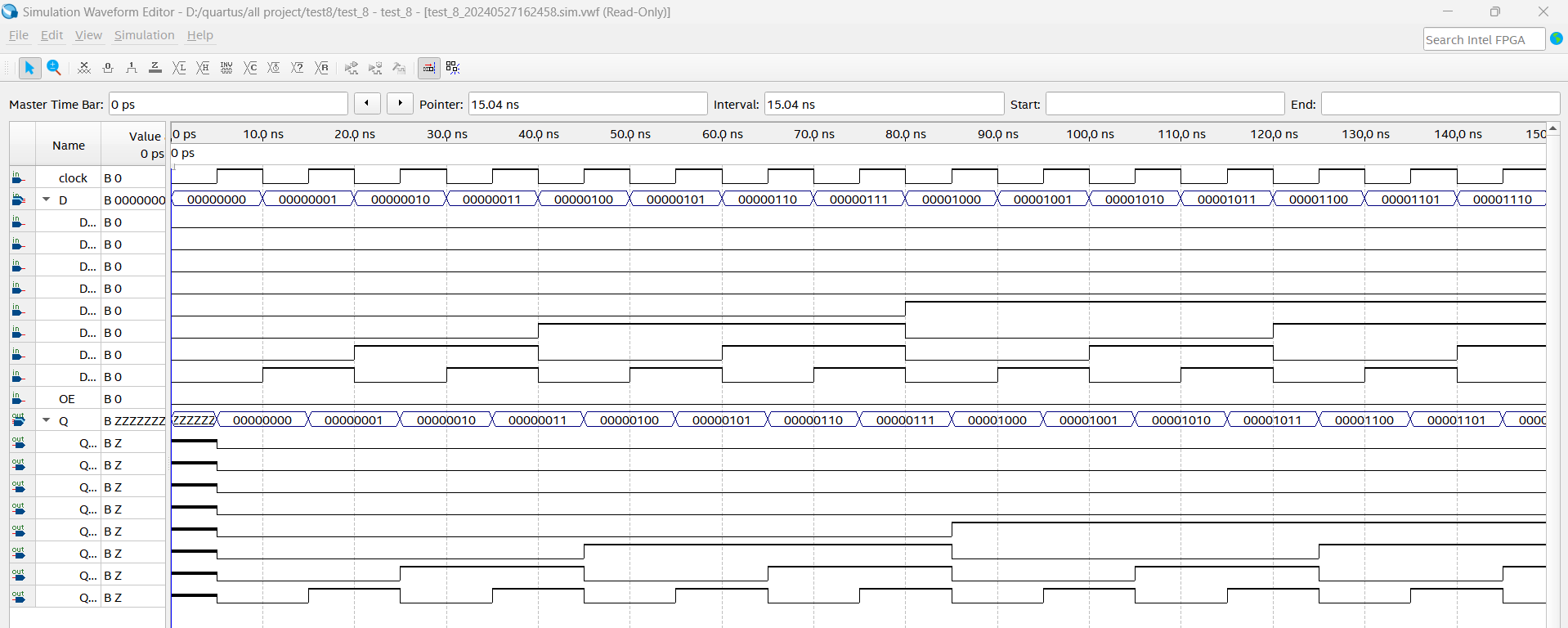
end if;

end process;

end architecture latch\_8;

解释：在OE为低电平时，根据D型触发器的状态转移方程Q=D，每当有一个上沿跳时，就将此刻D的状态赋值给Q。若OE为高电平，则显示高阻态。

4.波形图：



5.码上：

这段代码是用VHDL（Very High-Speed Integrated Circuit Hardware Description Language）编写的，用于描述数字电路的行为。它定义了一个名为test\_8的实体（entity），该实体具有三个输入端口clock、OE和D，以及一个输出端口Q。输入端口clock和OE都是1位宽的std\_logic，表示它们可以接收一个逻辑值（0或1）。输入端口D是一个8位宽的std\_logic\_vector，表示它可以接收八个逻辑值（0或1）。输出端口Q也是一个8位宽的std\_logic\_vector，用于输出结果。

在architecture部分，代码描述了实体的行为。具体来说，它使用了一个process语句来处理输入端口clock的变化，并根据这些变化更新输出端口Q的值。

①当OE为'0'时，执行以下操作：

如果clock发生上升沿（从0变为1），则将输入端口D的值赋给输出端口Q。

②当OE为'1'时，将输出端口Q的所有位都设置为高阻态（'Z'）。

总之，这段代码实现了一个简单的锁存器电路，根据输入端口clock和OE的值计算输出端口Q的值。当OE为'0'且clock发生上升沿时，锁存器会存储输入端口D的值；当OE为'1'时，锁存器的所有位都会处于高阻态。