实验报告

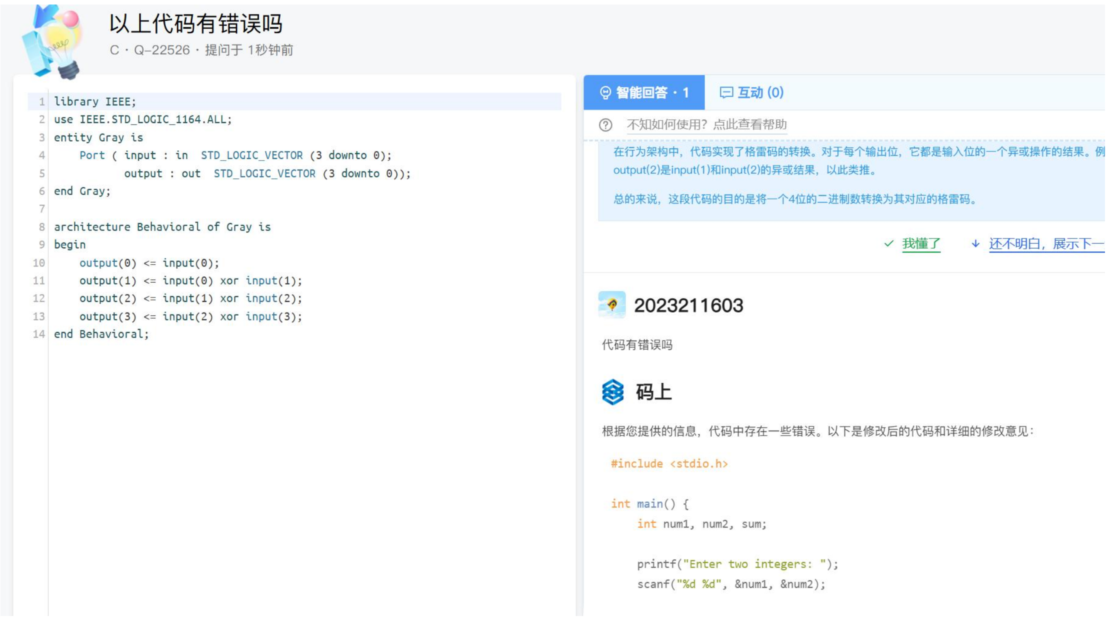
网安 5 班 王何佳 2023211603

王何佳

组合逻辑

实验一：8421 码和格雷码的转换

完成代码后上传“码上”进行纠错。



以上代码有错误吗

C · Q–22526 · 提问于 1秒钟前

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Gray is
    Port ( input : in  STD_LOGIC_VECTOR (3 downto 0);
           output : out  STD_LOGIC_VECTOR (3 downto 0));
end Gray;

architecture Behavioral of Gray is
begin
    output(0) <= input(0);
    output(1) <= input(0) xor input(1);
    output(2) <= input(1) xor input(2);
    output(3) <= input(2) xor input(3);
end Behavioral;
```

在行为架构中，代码实现了格雷码的转换。对于每个输出位，它都是输入位的一个异或操作的结果。例
output(2)是input(1)和input(2)的异或结果，以此类推。

总的来说，这段代码的目的是将一个4位的二进制数转换为其对应的格雷码。

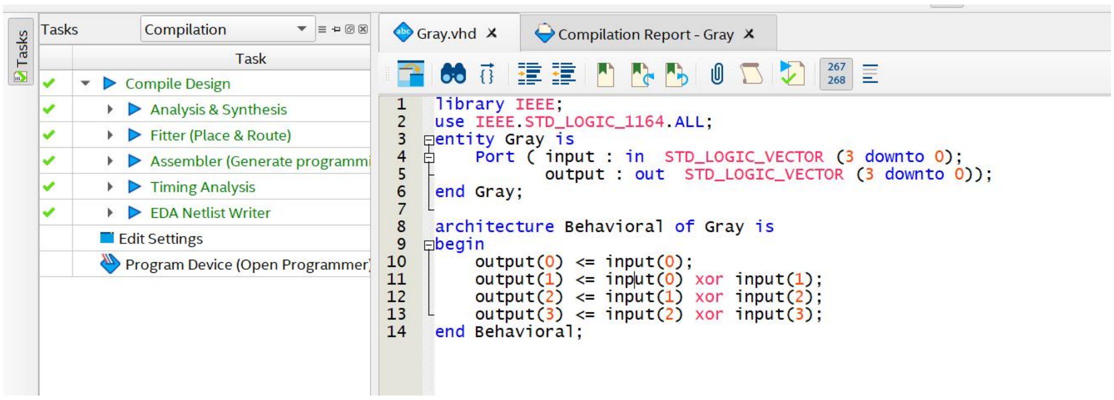
2023211603

代码有错误吗

码上

根据您提供的信息，代码中存在一些错误。以下是修改后的代码和详细的修改意见：

```
#include <stdio.h>

int main() {
    int num1, num2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);
```



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Gray is
    Port ( input : in  STD_LOGIC_VECTOR (3 downto 0);
           output : out  STD_LOGIC_VECTOR (3 downto 0));
end Gray;

architecture Behavioral of Gray is
begin
    output(0) <= input(0);
    output(1) <= input(0) xor input(1);
    output(2) <= input(1) xor input(2);
    output(3) <= input(2) xor input(3);
end Behavioral;
```

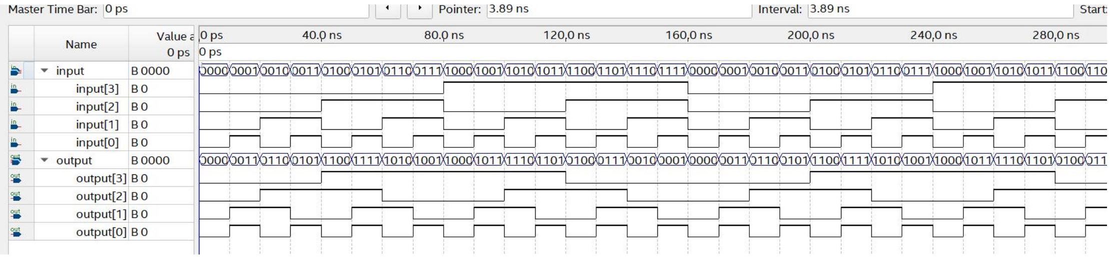
编译成功。

代码如下：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Gray is
    Port ( input : in    STD_LOGIC_VECTOR (3 downto 0);
               output : out    STD_LOGIC_VECTOR (3 downto 0));
end Gray;
```

```
architecture Behavioral of Gray is
begin
	output(0) <= input(0);
	output(1) <= input(0) xor input(1);
	output(2) <= input(1) xor input(2);
	output(3) <= input(2) xor input(3);
end Behavioral;
```

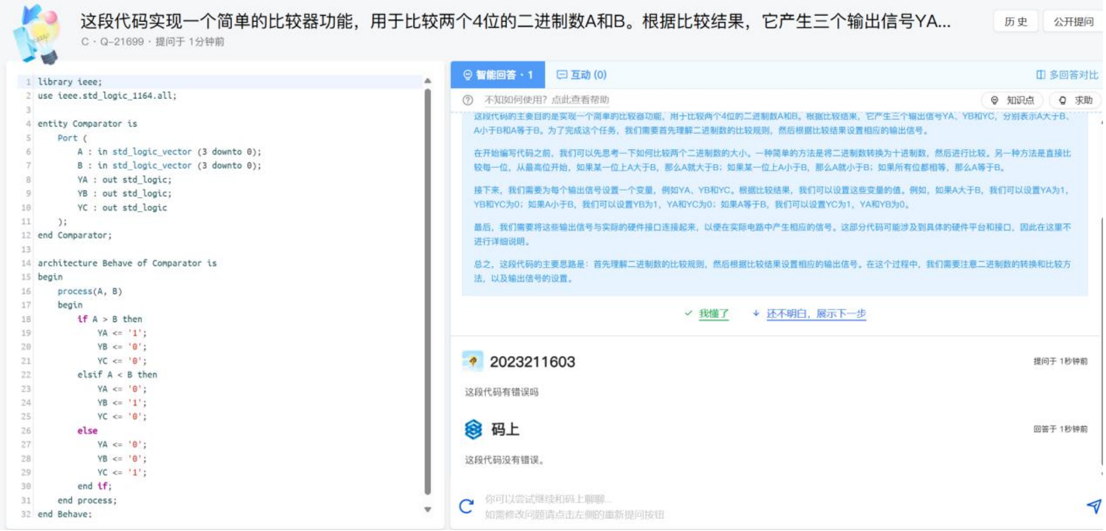
代码解释：以上代码实现了从 8421 码转为格雷码。
代码的主要部分包括一个实体定义和一个行为架构。实体定义了输入和输出端口，输入是一个 4 位的 std_logic_vector，输出也是一个 4 位的 std_logic_vector。
在行为架构中，代码实现了格雷码的转换。对于每个输出位，它都是输入位的一个异或操作的结果。例如，output(1)是 input(0)和 input(1)的异或结果，output(2)是 input(1)和 input(2)的异或结果，以此类推。
波形图：



波形图中正常将输入的 8421 码转换为格雷码。

实验二：数值比较器

完成代码后上传“码上”进行纠错。



这段代码实现一个简单的比较器功能，用于比较两个4位的二进制数A和B。根据比较结果，它产生三个输出信号YA...

C · Q-21699 · 提问于 1分钟前

```
library ieee;
use ieee.std_logic_1164.all;

entity Comparator is
    Port (
        A : in std_logic_vector (3 downto 0);
        B : in std_logic_vector (3 downto 0);
        YA : out std_logic;
        YB : out std_logic;
        YC : out std_logic
    );
end Comparator;

architecture Behave of Comparator 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 Behave;
```

这段代码的主要目的是实现一个简单的比较器功能，用于比较两个4位的二进制数A和B。根据比较结果，它产生三个输出信号YA、YB和YC，分别表示A大于B、A小于B和A等于B。为了完成这个任务，我们需要首先理解二进制数的比较规则，然后根据比较结果设置相应的输出信号。

在开始编写代码之前，我们可以先思考一下如何比较两个二进制数的大小。一种简单的方法是将二进制数转换为十进制数，然后进行比较。另一种方法是直接比较每一位，从最高位开始，如果某一位上A大于B，那么A就大于B；如果某一位上A小于B，那么A就小于B；如果所有位都相等，那么A等于B。

接下来，我们需要为每个输出信号设置一个变量，例如YA、YB和YC。根据比较结果，我们可以设置这些变量的值。例如，如果A大于B，我们可以设置YA为1，YB和YC为0；如果A小于B，我们可以设置YB为1，YA和YC为0；如果A等于B，我们可以设置YC为1，YA和YB为0。

最后，我们需要将这些输出信号与实际的硬件接口连接起来，以便在实际电路中产生相应的信号。这部分代码可能涉及到具体的硬件平台和接口，因此在这里不进行详细说明。

总之，这段代码的主要思路是：首先理解二进制数的比较规则，然后根据比较结果设置相应的输出信号。在这个过程中，我们需要注意二进制数的转换和比较方法，以及输出信号的设置。

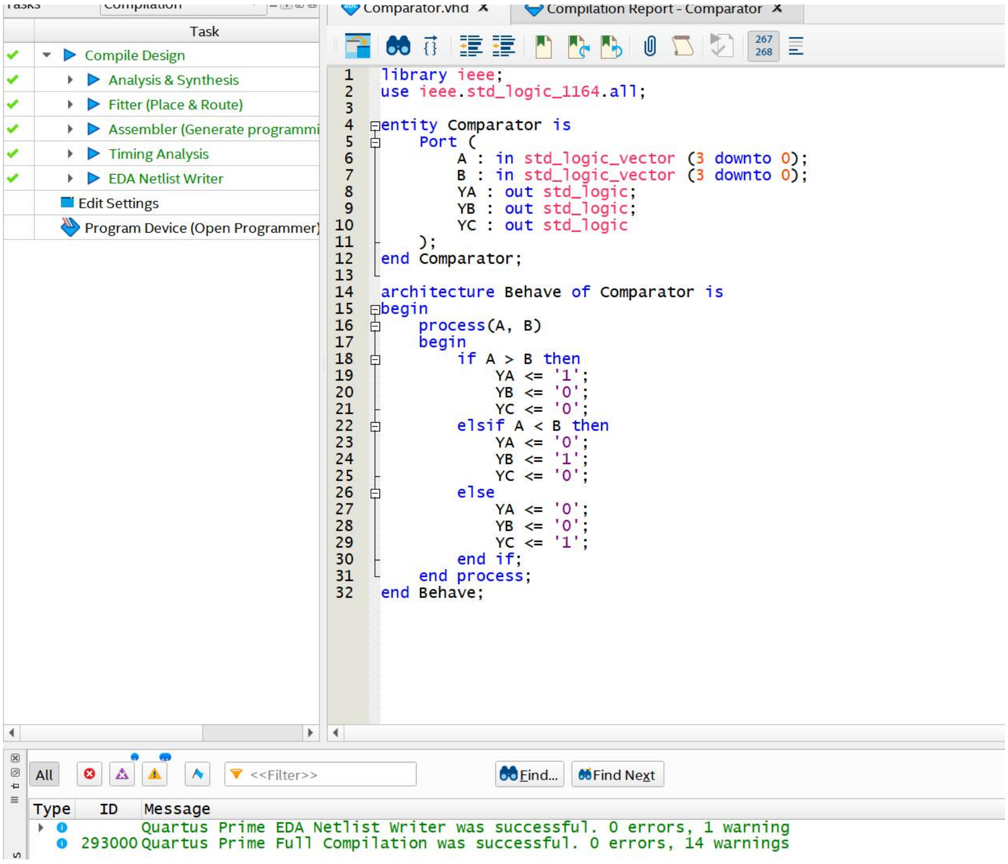
2023211603 提问于 1秒钟前

这段代码有错误吗

码上 回答于 1秒钟前

这段代码没有错误。

你可以尝试继续和码上聊聊...
如需修改问题请点击左侧的重新提问按钮



```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity Comparator is
    Port (
        A : in std_logic_vector (3 downto 0);
        B : in std_logic_vector (3 downto 0);
        YA : out std_logic;
        YB : out std_logic;
        YC : out std_logic
    );
end Comparator;

architecture Behave of Comparator 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 Behave;
```

```
       Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings
```

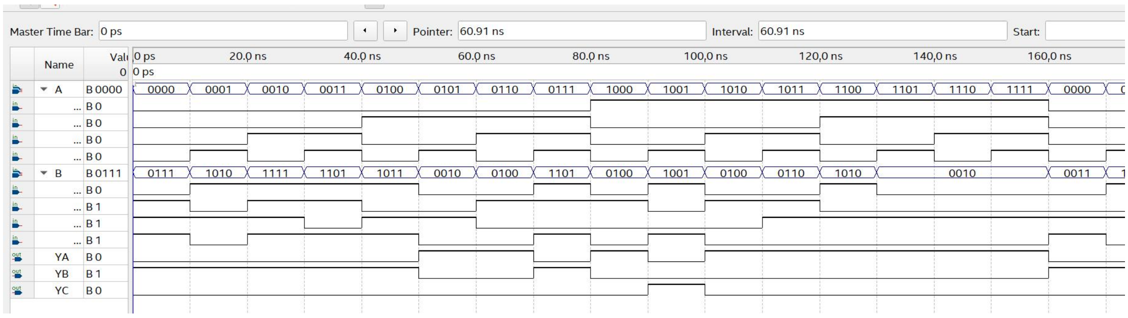
编译成功。

代码如下：

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity Comparator is
    Port (
        A : in std_logic_vector (3 downto 0);
        B : in std_logic_vector (3 downto 0);
        YA : out std_logic;
        YB : out std_logic;
        YC : out std_logic
    );
end Comparator;

architecture Behave of Comparator 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 Behave;
```

代码解释：这段代码实现了一个简单的比较器功能，用于比较两个 4 位的二进制数 A 和 B。根据比较结果，它产生三个输出信号 YA、YB 和 YC，分别表示 A 大于 B、A 小于 B 和 A 等于 B。

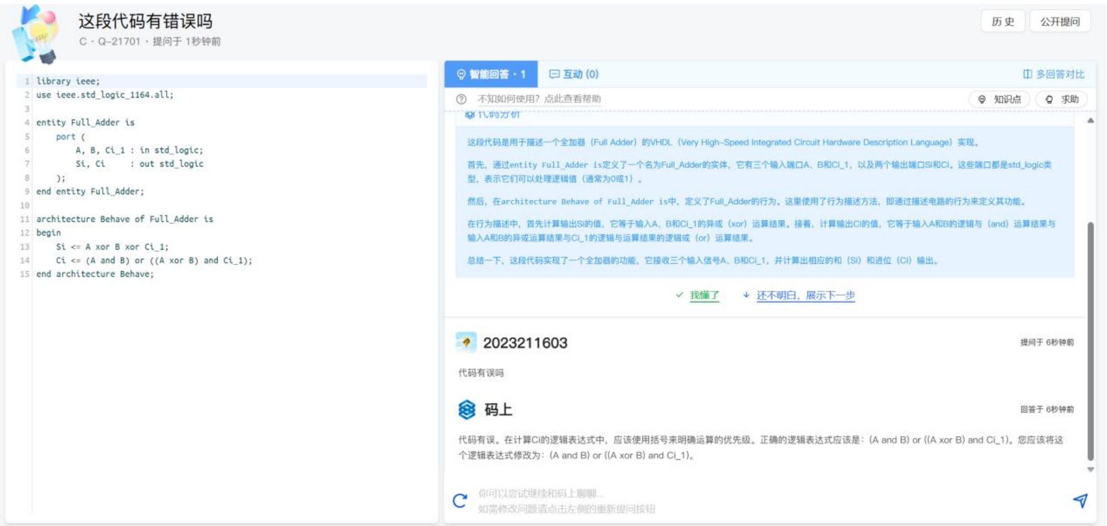
波形图：



A 输入设置为 0000 开始递增 1，B 输入设置为随机数，输出结果符合 A>B 时 YA 为高电平，A<B 时 YB 为低电平，A=B 时 YC 为高电平的要求。

实验三：全加器

完成代码后上传“码上”进行纠错。



这段代码有错误吗

C · Q-21701 · 提问于 1秒钟前

```
library ieee;
use ieee.std_logic_1164.all;

entity Full_Adder is
    port (
        A, B, Ci_1 : in std_logic;
        Si, Ci     : out std_logic
    );
end entity Full_Adder;

architecture Behave of Full_Adder is
begin
    Si <= A xor B xor Ci_1;
    Ci <= (A and B) or ((A xor B) and Ci_1);
end architecture Behave;
```

这段代码是用于描述一个全加器（Full Adder）的VHDL（Very High-Speed Integrated Circuit Hardware Description Language）实现。

首先，通过`entity Full_Adder is`定义了一个名为Full_Adder的实体，它有三个输入端口A、B和Ci_1，以及两个输出端口Si和Ci。这些端口都是std_logic类型，表示它们可以处理逻辑值（通常为0或1）。

然后，在`architecture Behave of Full_Adder is`中，定义了Full_Adder的行为。这里使用了行为描述方法，即通过描述电路的行为来定义其功能。

在行为描述中，首先计算输出Si的值，它等于输入A、B和Ci_1的异或（xor）运算结果。接着，计算输出Ci的值，它等于输入A和B的逻辑与（and）运算结果与输入A和B的异或运算结果与Ci_1的逻辑与运算结果的逻辑或（or）运算结果。

总结一下，这段代码实现了一个全加器的功能，它接收三个输入信号A、B和Ci_1，并计算出相应的和（Si）和进位（Ci）输出。

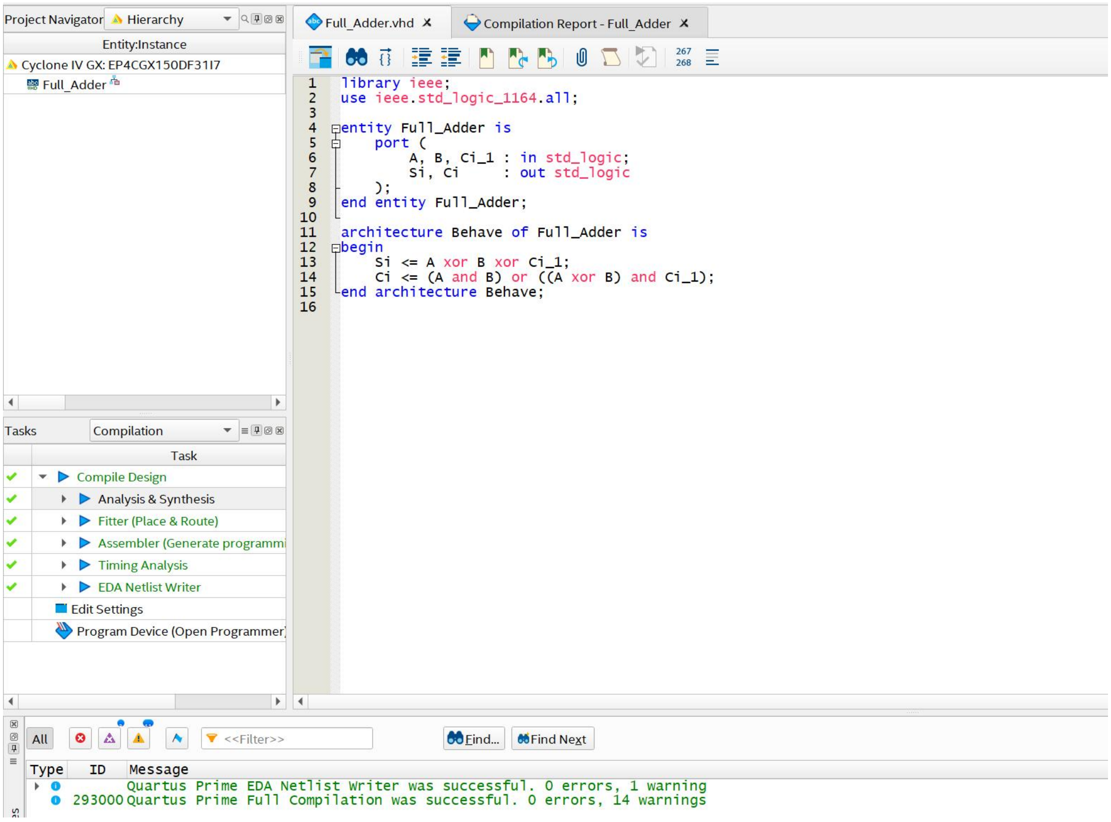
2023211603 提问于 6秒钟前

代码有误吗

码上 回答于 6秒钟前

代码有误。在计算Ci的逻辑表达式中，应该使用括号来明确运算的优先级。正确的逻辑表达式应该是：(A and B) or ((A xor B) and Ci_1)。您应该将这个逻辑表达式修改为：(A and B) or ((A xor B) and Ci_1)。

你可以尝试继续和码上聊聊...
如需修改问题请点击左侧的重新提问按钮



```
library ieee;
use ieee.std_logic_1164.all;

entity Full_Adder is
    port (
        A, B, Ci_1 : in std_logic;
        Si, Ci     : out std_logic
    );
end entity Full_Adder;

architecture Behave of Full_Adder is
begin
    Si <= A xor B xor Ci_1;
    Ci <= (A and B) or ((A xor B) and Ci_1);
end architecture Behave;
```

```
Type   ID     Message
              Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
       293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings
```

编译成功

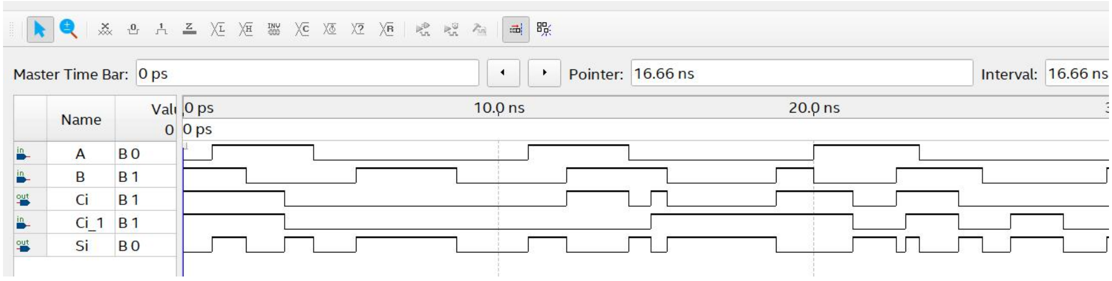
代码如下：

```
library ieee;
use ieee.std_logic_1164.all;

entity Full_Adder is
	port (
		A, B, Ci_1 : in std_logic;
		Si, Ci       : out std_logic
	);
end entity Full_Adder;

architecture Behave of Full_Adder is
begin
	Si <= A xor B xor Ci_1;
	Ci <= (A and B) or ((A xor B) and Ci_1);
end architecture Behave;
```

代码解释：“Si <= A xor B xor Ci_1”实现了求和逻辑。Si 是 A、B 和 Ci_1 通过异或（XOR）操作的结果。异或操作的特性是，当输入的两个位相同时输出为 0，不同则为 1。因此，这条语句计算了 A 和 B 相加的结果，再加上进位输入 Ci_1。

Ci <= (A and B) or ((A xor B) and Ci_1)实现了进位输出逻辑。Ci 是两个条件的逻辑或（OR）结果。第一个条件是 A 和 B 都为 1（即它们相加产生了进位）。第二个条件是 A 和 B 不同，且 Ci_1 为 1（即有一个进位输入，加上 A 和 B 的异或结果也产生了进位）。这确保了任何时候只要有进位产生，Ci 就会被设置为 1。

波形图：

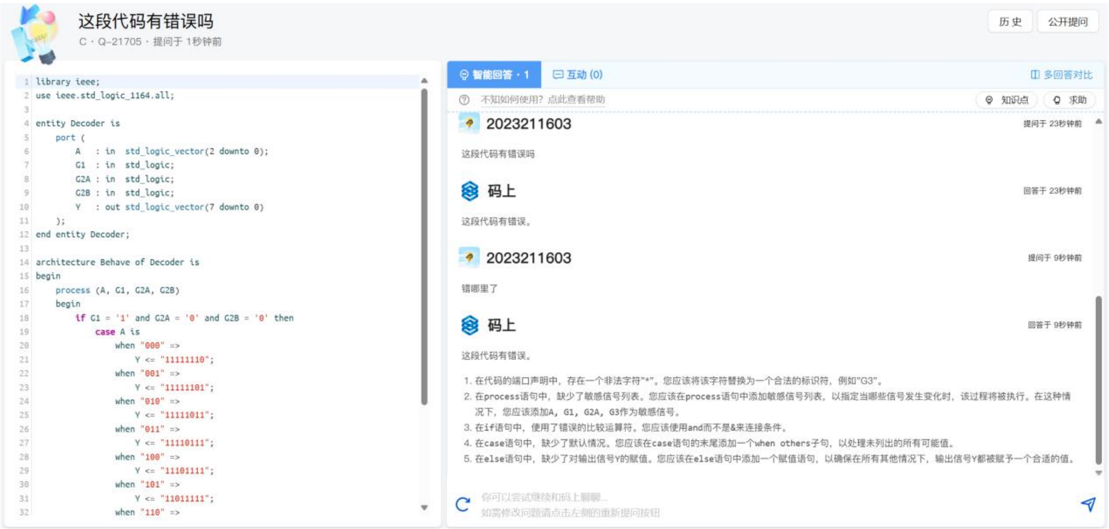

Master Time Bar: 0 ps
Pointer: 16.66 ns
Interval: 16.66 ns

Name	Value
A	B 0
B	B 1
Ci	B 1
Ci_1	B 1
Si	B 0

A,B,Ci_1 均设为随机时间间隔，观察输出波形，满足 Si=Ai⊕Bi⊕Ci-1，Ci=(Ai⊕Bi)Ci-1+ AiBi。

实验四 3 线-8 线译码器

完成代码后上传“码上”进行纠错。


这段代码有错误吗

C · Q-21705 · 提问于 1秒钟前

```
library ieee;
use ieee.std_logic_1164.all;

entity Decoder is
    port (
        A   : in  std_logic_vector(2 downto 0);
        G1  : in  std_logic;
        G2A : in  std_logic;
        G2B : in  std_logic;
        Y   : out std_logic_vector(7 downto 0)
    );
end entity Decoder;

architecture Behave of Decoder is
begin
    process (A, G1, G2A, G2B)
    begin
        if G1 = '1' and G2A = '0' and G2B = '0' then
            case A is
                when "000" =>
                    Y <= "11111110";
                when "001" =>
                    Y <= "11111101";
                when "010" =>
                    Y <= "11111011";
                when "011" =>
                    Y <= "11110111";
                when "100" =>
                    Y <= "11101111";
                when "101" =>
                    Y <= "11011111";
                when "110" =>
```

2023211603 提问于 23秒钟前

这段代码有错误吗

码上 回答于 23秒钟前

这段代码有错误。

2023211603 提问于 9秒钟前

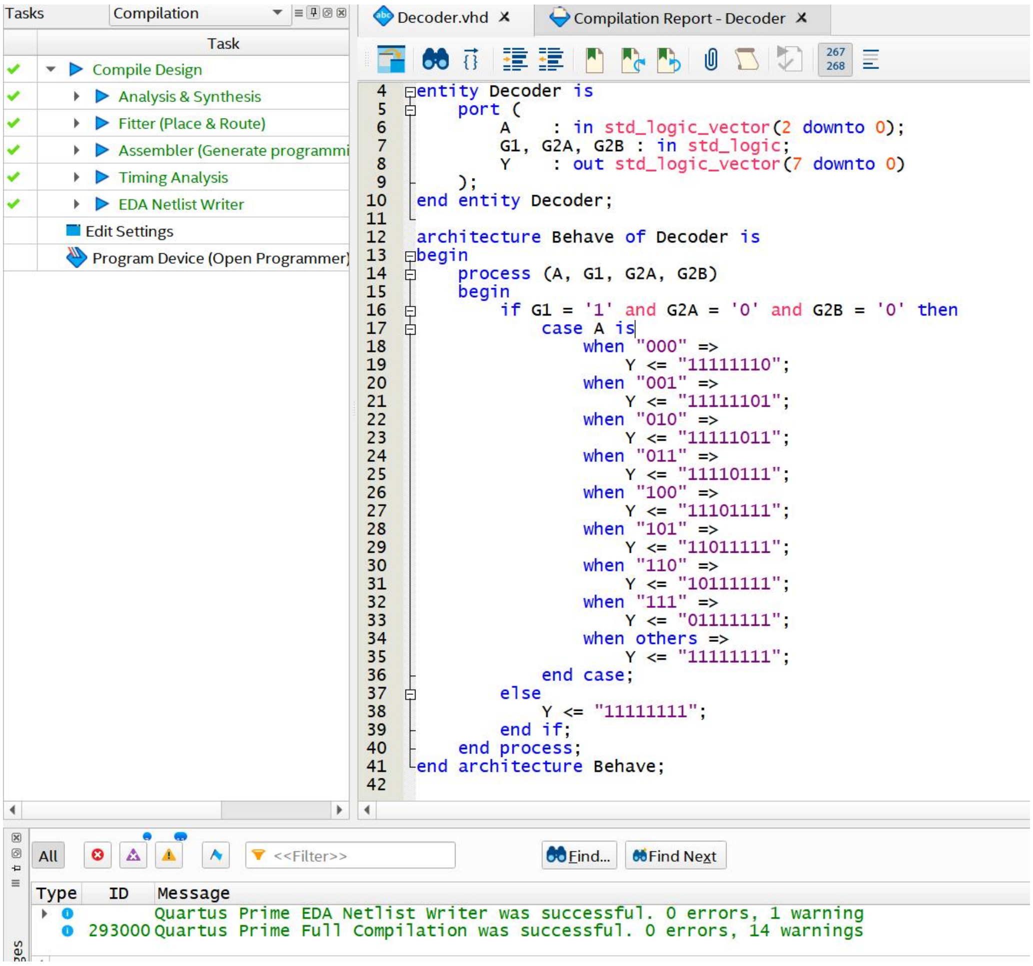
错哪里了

码上 回答于 9秒钟前

这段代码有错误。

1. 在代码的端口声明中，存在一个非法字符"*"。您应该将该字符替换为一个合法的标识符，例如"G3"。
2. 在process语句中，缺少了敏感信号列表。您应该在process语句中添加敏感信号列表，以指定当哪些信号发生变化时，该过程将被执行。在这种情况下，您应该添加A，G1，G2A，G3作为敏感信号。
3. 在if语句中，使用了错误的比较运算符。您应该使用and而不是&来连接条件。
4. 在case语句中，缺少了默认情况。您应该在case语句的末尾添加一个when others子句，以处理未列出的所有可能值。
5. 在else语句中，缺少了对输出信号Y的赋值。您应该在else语句中添加一个赋值语句，以确保在所有其他情况下，输出信号Y都被赋予一个合适的值。

历 史
公开提问
智能回答 · 1
互动 (0)
多回答对比
不知如何使用？点此查看帮助
知识点
求助
你可以尝试继续和码上聊聊...
如需修改问题请点击左侧的重新提问按钮



```
entity Decoder is
    port (
        A    : in std_logic_vector(2 downto 0);
        G1, G2A, G2B : in std_logic;
        Y    : out std_logic_vector(7 downto 0)
    );
end entity Decoder;

architecture Behave of Decoder is
begin
    process (A, G1, G2A, G2B)
    begin
        if G1 = '1' and G2A = '0' and G2B = '0' then
            case A is
                when "000" =>
                    Y <= "11111110";
                when "001" =>
                    Y <= "11111101";
                when "010" =>
                    Y <= "11111011";
                when "011" =>
                    Y <= "11110111";
                when "100" =>
                    Y <= "11101111";
                when "101" =>
                    Y <= "11011111";
                when "110" =>
                    Y <= "10111111";
                when "111" =>
                    Y <= "01111111";
                when others =>
                    Y <= "11111111";
            end case;
        else
            Y <= "11111111";
        end if;
    end process;
end architecture Behave;
```

```
Type   ID     Message
             Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
       293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings
```

编译成功。

代码如下：

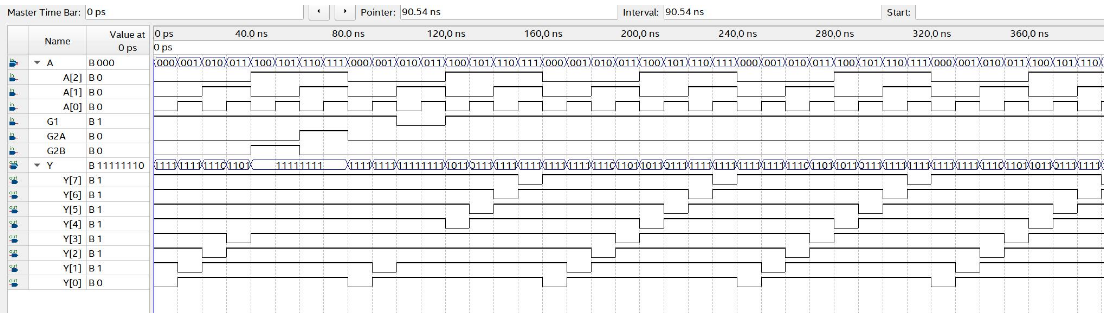
```
library ieee;
use ieee.std_logic_1164.all;

entity Decoder is
    port (
        A     : in    std_logic_vector(2 downto 0);
        G1    : in    std_logic;
        G2A : in    std_logic;
        G2B : in    std_logic;
        Y     : out std_logic_vector(7 downto 0)
    );
end entity Decoder;

architecture Behave of Decoder is
begin
    process (A, G1, G2A, G2B)
    begin
        if G1 = '1' and G2A = '0' and G2B = '0' then
```

```
            case A is
                when "000" =>
                    Y <= "11111110";
                when "001" =>
                    Y <= "11111101";
                when "010" =>
                    Y <= "11111011";
                when "011" =>
                    Y <= "11110111";
                when "100" =>
                    Y <= "11101111";
                when "101" =>
                    Y <= "11011111";
                when "110" =>
                    Y <= "10111111";
                when "111" =>
                    Y <= "01111111";
            end case;
        else
            Y <= "11111111";
        end if;
    end process;
end architecture Behave;
```

代码解释：

Decoder 实体中定义了 3 位逻辑变量输入 A 和使能端输入 G1,G2A,G2B 以及 8 位逻辑向量输 出 Y，Behave 中使用 if 和 case 语句，使得仅 G1 为高电平且 G2A，G2B 为低电平时器件才正常工作。

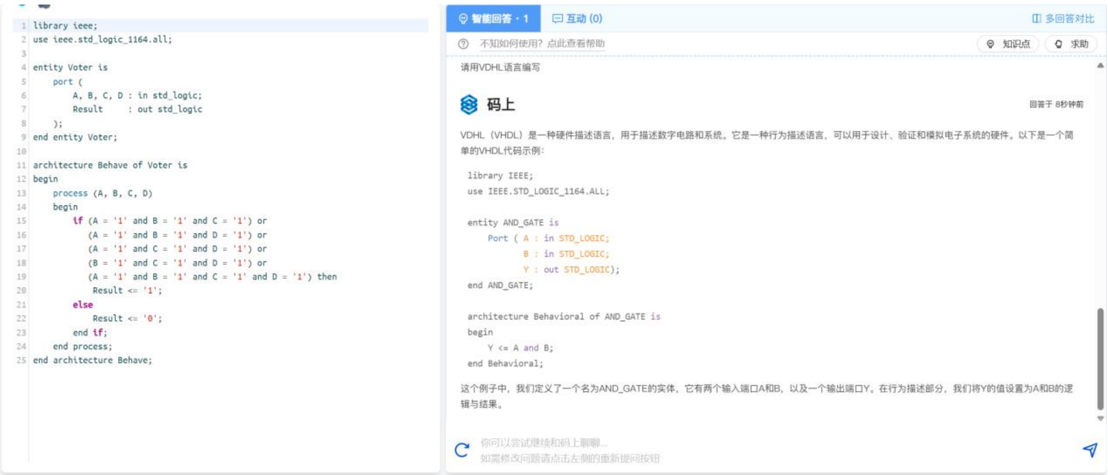
波形图：



上图中当 G1 为低电平或 G2 为高电平或 G3 为高电平时无效，输出 11111111。其他正常工作状态下，将 3 线输入转化进行输出。

实验五：表决器

完成代码后上传“码上”进行纠错。



```
library ieee;
use ieee.std_logic_1164.all;

entity Voter is
    port (
        A, B, C, D : in std_logic;
        Result     : out std_logic
    );
end entity Voter;

architecture Behave of Voter is
begin
    process (A, B, C, D)
    begin
        if (A = '1' and B = '1' and C = '1') or
           (A = '1' and B = '1' and D = '1') or
           (A = '1' and C = '1' and D = '1') or
           (B = '1' and C = '1' and D = '1') or
           (A = '1' and B = '1' and C = '1' and D = '1') then
            Result <= '1';
        else
            Result <= '0';
        end if;
    end process;
end architecture Behave;
```

请用VDHL语言编写

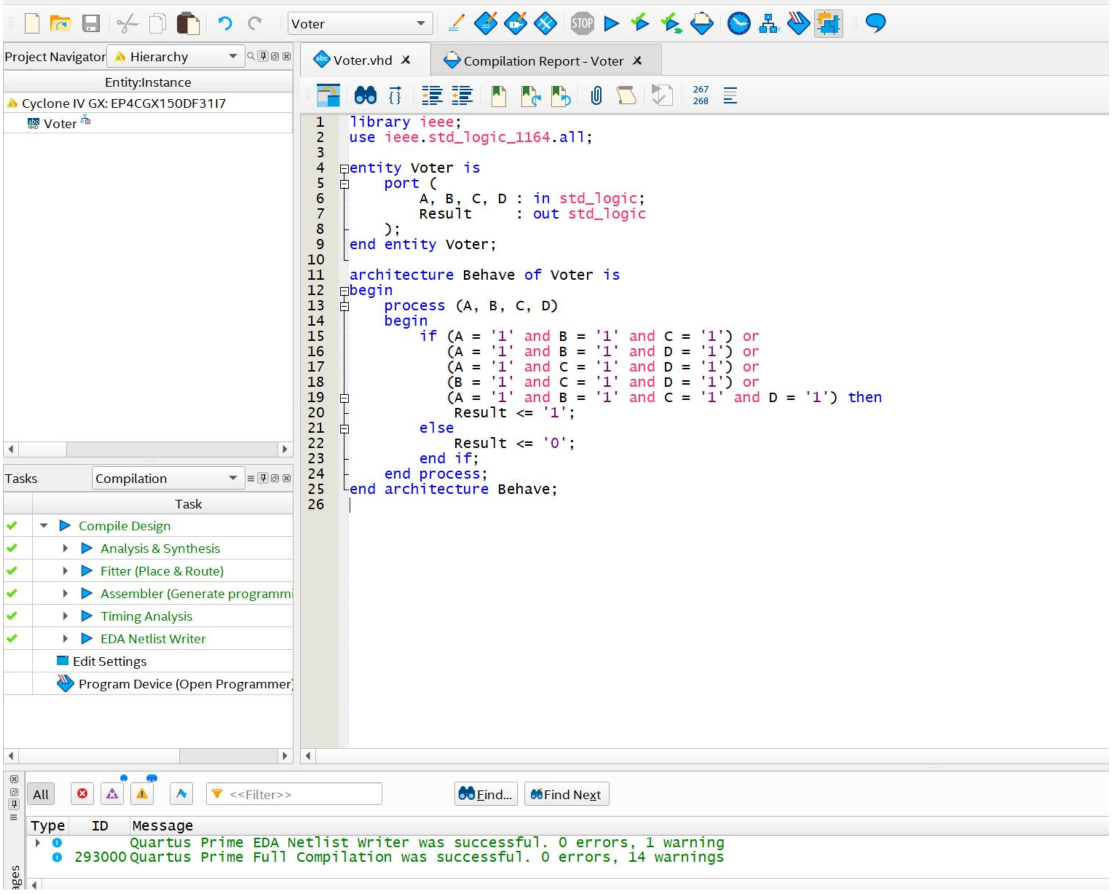
码上

VDHL（VHDL）是一种硬件描述语言，用于描述数字电路和系统。它是一种行为描述语言，可以用于设计、验证和模拟电子系统的硬件。以下是一个简单的VHDL代码示例：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AND_GATE is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Y : out STD_LOGIC);
end AND_GATE;

architecture Behavioral of AND_GATE is
begin
    Y <= A and B;
end Behavioral;
```

这个例子中，我们定义了一个名为AND_GATE的实体，它有两个输入端口A和B，以及一个输出端口Y。在行为描述部分，我们将Y的值设置为A和B的逻辑与结果。

你可以尝试继续和码上聊聊...
如需修改问题请点击左侧的重新提问按钮



```
library ieee;
use ieee.std_logic_1164.all;

entity Voter is
    port (
        A, B, C, D : in std_logic;
        Result     : out std_logic
    );
end entity Voter;

architecture Behave of Voter is
begin
    process (A, B, C, D)
    begin
        if (A = '1' and B = '1' and C = '1') or
           (A = '1' and B = '1' and D = '1') or
           (A = '1' and C = '1' and D = '1') or
           (B = '1' and C = '1' and D = '1') or
           (A = '1' and B = '1' and C = '1' and D = '1') then
            Result <= '1';
        else
            Result <= '0';
        end if;
    end process;
end architecture Behave;
```

```
Type   ID     Message
              Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
       293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings
```

编译成功。

代码如下：

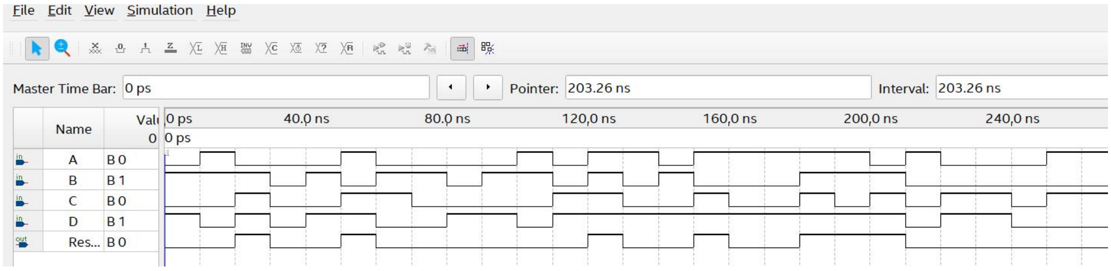
```
library ieee;
use ieee.std_logic_1164.all;

entity Voter is
	port (
		A, B, C, D : in std_logic;
		Result		: out std_logic
```

```
        );
end entity Voter;

architecture Behave of Voter is
begin
        process (A, B, C, D)
        begin
                if (A = '1' and B = '1' and C = '1') or
                      (A = '1' and B = '1' and D = '1') or
                      (A = '1' and C = '1' and D = '1') or
                      (B = '1' and C = '1' and D = '1') or
                      (A = '1' and B = '1' and C = '1' and D = '1') then
                        Result <= '1';
                else
                        Result <= '0';
                end if;
        end process;
end architecture Behave;
```

代码解释：

这段代码实现了一个表决器，它可以接收四个输入信号 A、B、C 和 D，并根据多数原则产生一个输出 Result。如果至少有三个输入为'1'，输出将为'1'；否则，输出为'0'。、

波形图：



Master Time Bar: 0 ps
Pointer: 203.26 ns
Interval: 203.26 ns

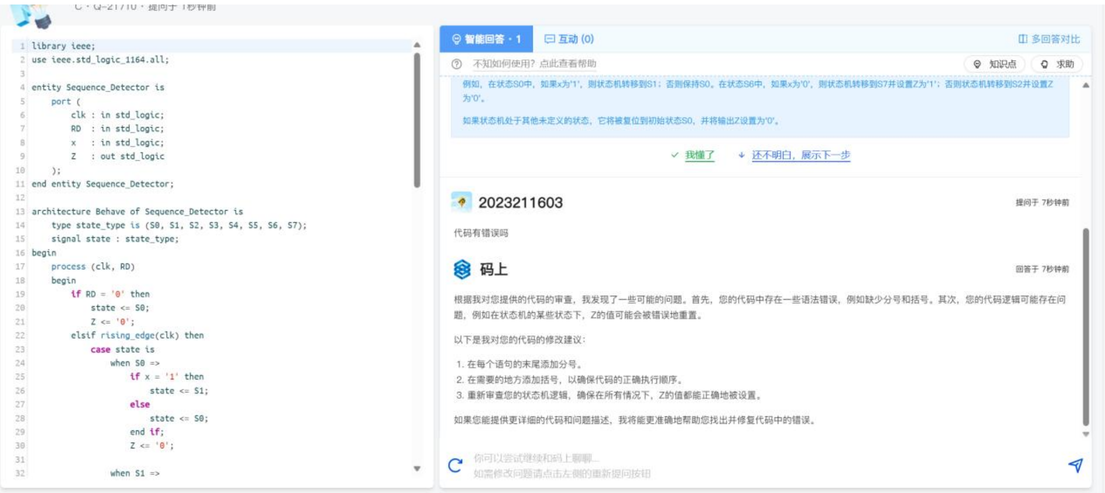
Name	Value at 0 ps
A	B 0
B	B 1
C	B 0
D	B 1
Res...	B 0

从波形图可以看出，当有 2 个以上输入为高电平时，输出为高电平，否则为低电平。

时序逻辑

实验一：序列检测器

完成代码后上传“码上”进行纠错。



```
library ieee;
use ieee.std_logic_1164.all;

entity Sequence_Detector is
    port (
        clk : in std_logic;
        RD  : in std_logic;
        x   : in std_logic;
        Z   : out std_logic
    );
end entity Sequence_Detector;

architecture Behave of Sequence_Detector is
    type state_type is (S0, S1, S2, S3, S4, S5, S6, S7);
    signal state : state_type;
begin
    process (clk, RD)
    begin
        if RD = '0' then
            state <= S0;
            Z <= '0';
        elsif rising_edge(clk) then
            case state is
                when S0 =>
                    if x = '1' then
                        state <= S1;
                    else
                        state <= S0;
                    end if;
                    Z <= '0';

                when S1 =>
```

例如，在状态S0中，如果x为'1'，则状态机转移到S1；否则保持S0。在状态S6中，如果x为'0'，则状态机转移到S7并设置Z为'1'；否则状态机转移到S2并设置Z为'0'。

如果状态机处于其他未定义的状态，它将被复位到初始状态S0，并将输出Z设置为'0'。

2023211603 提问于 7秒钟前

代码有错误吗

码上 回答于 7秒钟前

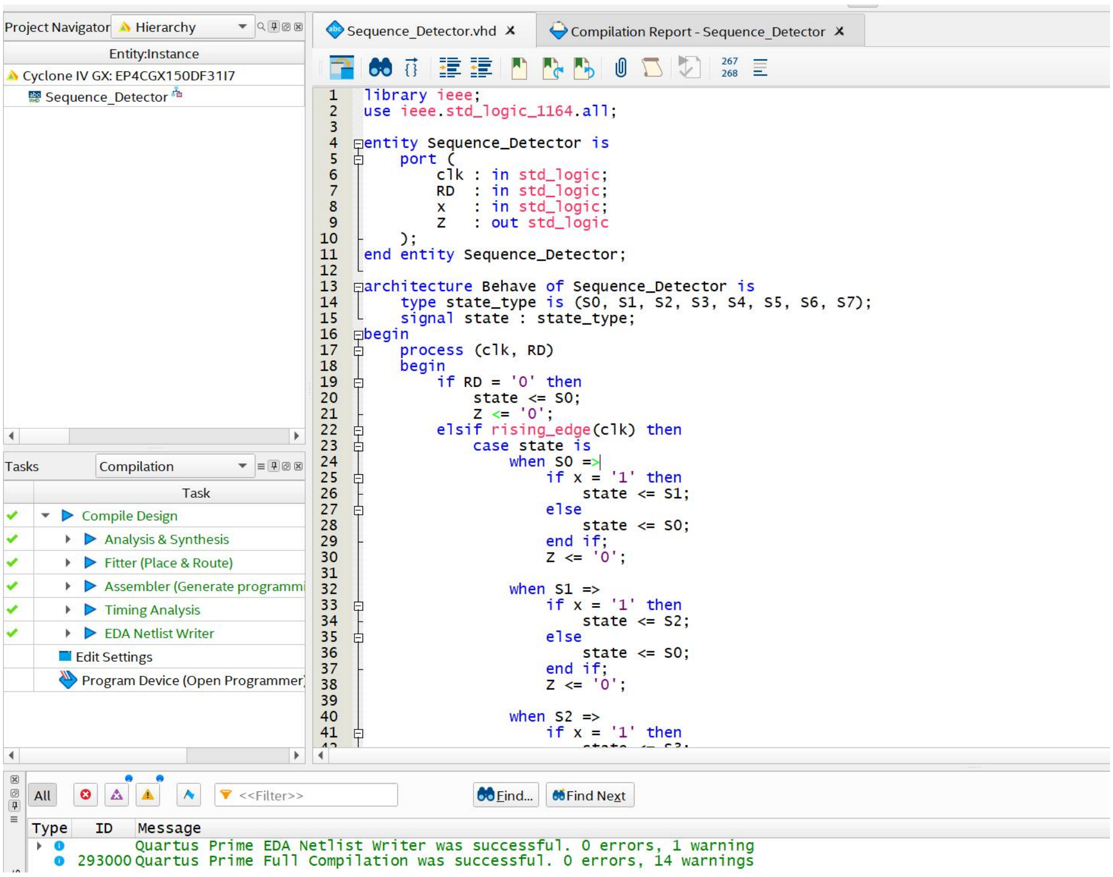
根据我对您提供的代码的审查，我发现了一些可能的问题。首先，您的代码中存在一些语法错误，例如缺少分号和括号。其次，您的代码逻辑可能存在问题，例如在状态机的某些状态下，Z的值可能会被错误地重置。

以下是我对您的代码的修改建议：

1. 在每个语句的末尾添加分号。
2. 在需要的地方添加括号，以确保代码的正确执行顺序。
3. 重新审查您的状态机逻辑，确保在所有情况下，Z的值都能正确地被设置。

如果您能提供更详细的代码和问题描述，我将能更准确地帮助您找出并修复代码中的错误。

你可以尝试继续和码上聊聊...
如需修改问题请点击左侧的重新提问按钮



```
library ieee;
use ieee.std_logic_1164.all;

entity Sequence_Detector is
    port (
        clk  :  in std_logic;
        RD   :  in std_logic;
        x    :  in std_logic;
        Z    :  out std_logic
    );
end entity Sequence_Detector;

architecture Behave of Sequence_Detector is
    type state_type is (S0, S1, S2, S3, S4, S5, S6, S7);
    signal state : state_type;
begin
    process (clk, RD)
    begin
        if RD = '0' then
            state <= S0;
            Z <= '0';
        elsif rising_edge(clk) then
            case state is
                when S0 =>
                    if x = '1' then
                        state <= S1;
                    else
                        state <= S0;
                    end if;
                    Z <= '0';

                when S1 =>
                    if x = '1' then
                        state <= S2;
                    else
                        state <= S0;
                    end if;
                    Z <= '0';

                when S2 =>
                    if x = '1' then
```

```
Type   ID     Message
                Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
         293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings
```

编译成功。

代码如下：

library ieee;

use ieee.std_logic_1164.all;

entity Sequence_Detector is

```
        port (
                clk : in std_logic;
                RD    : in std_logic;
                x      : in std_logic;
                Z      : out std_logic
        );
end entity Sequence_Detector;

architecture Behave of Sequence_Detector is
        type state_type is (S0, S1, S2, S3, S4, S5, S6, S7);
        signal state : state_type;
begin
        process (clk, RD)
        begin
                if RD = '0' then
                        state <= S0;
                        Z <= '0';
                elsif rising_edge(clk) then
                        case state is
                                when S0 =>
                                        if x = '1' then
                                                state <= S1;
                                        else
                                                state <= S0;
                                        end if;
                                        Z <= '0';

                                when S1 =>
                                        if x = '1' then
                                                state <= S2;
                                        else
                                                state <= S0;
                                        end if;
                                        Z <= '0';

                                when S2 =>
                                        if x = '1' then
                                                state <= S3;
                                        else
                                                state <= S0;
                                        end if;
                                        Z <= '0';

                                when S3 =>
```

```
    if x = '0' then
        state <= S4;
    else
        state <= S3;
    end if;
    Z <= '0';

when S4 =>
    if x = '0' then
        state <= S5;
    else
        state <= S0;
    end if;
    Z <= '0';

when S5 =>
    if x = '1' then
        state <= S6;
    else
        state <= S0;
    end if;
    Z <= '0';

when S6 =>
    if x = '0' then
        state <= S7;
        Z <= '1';
    else
        state <= S2;
        Z <= '0';
    end if;

when S7 =>
    if x = '1' then
        state <= S0;
        Z <= '1';
    else
        state <= S0;
        Z <= '0';
    end if;

when others =>
    state <= S0;
    Z <= '0';
```

```
                end case;
            end if;
        end process;
end architecture Behave;
```

代码解释：

port: 定义了 Sequence_Detector 的接口。

clk: 输入端口，类型为 std_logic，表示时钟信号。

RD: 输入端口，类型为 std_logic，表示复位信号。

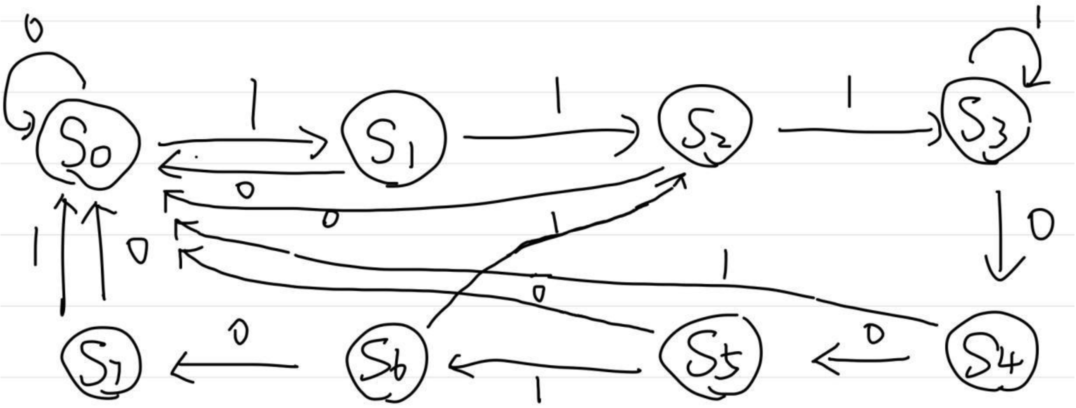
x: 输入端口，类型为 std_logic，表示序列输入信号。

Z: 输出端口，类型为 std_logic，表示检测到序列时的输出信号。

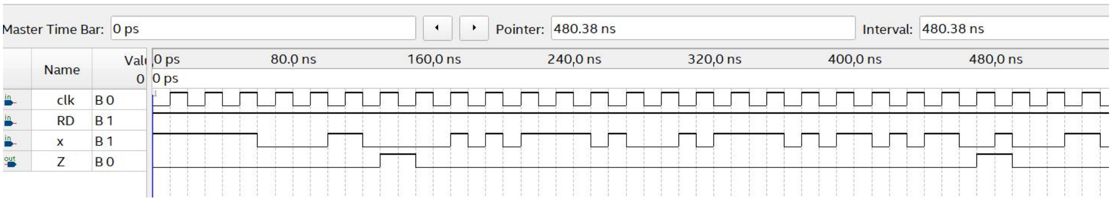
当复位信号 RD 为'0'时，将状态重置为初始状态 S0，并将输出 Z 置为'0'。

当时钟信号上升沿发生且复位信号 RD 不为'0'时，根据当前状态和输入信号 x 来决定下一个状态。当输入出现连续的 1110010 时，输出 Z 为 1，否则为 0。

状态转移图：

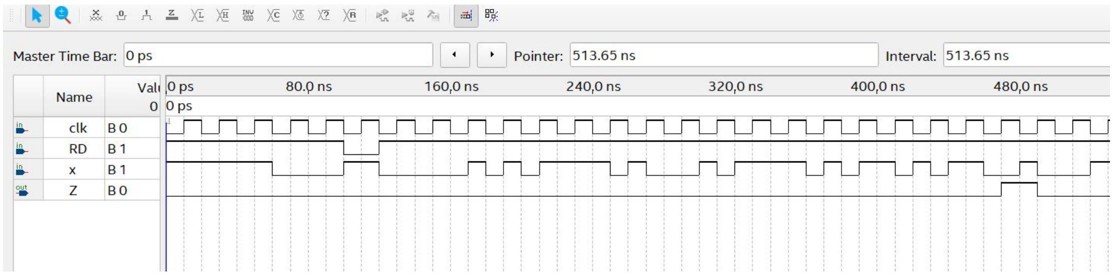


波形图：



Master Time Bar: 0 ps
Pointer: 480.38 ns
Interval: 480.38 ns

当 RD 一直处于高电平时，序列检测器正常工作，当侦测到连续的 1110010 信号时，Z 将输出 1。

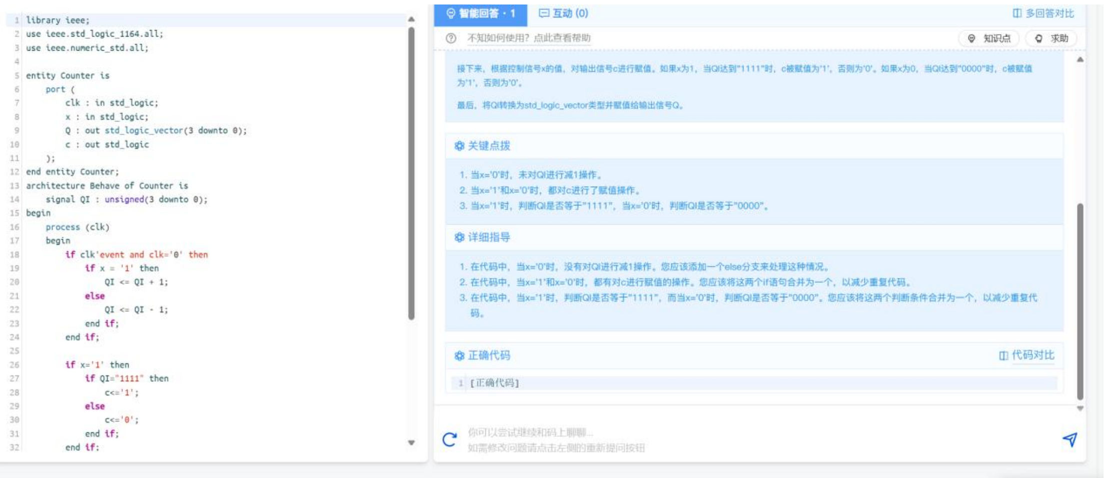


Master Time Bar: 0 ps
Pointer: 513.65 ns
Interval: 513.65 ns

若保持上面的输入不变，在第一个 1110010 序列内插入低电平 RD 信号使状态复位，可以看到第一个序列后 Z 并没有变为高电平。

实验二：计数器

完成代码后上传“码上”进行纠错。



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Counter is
    port (
        clk : in std_logic;
        x : in std_logic;
        Q : out std_logic_vector(3 downto 0);
        c : out std_logic
    );
end entity Counter;
architecture Behave of Counter is
    signal QI : unsigned(3 downto 0);
begin
    process (clk)
    begin
        if clk'event and clk='0' then
            if x = '1' then
                QI <= QI + 1;
            else
                QI <= QI - 1;
            end if;
        end if;

        if x='1' then
            if QI="1111" then
                c<='1';
            else
                c<='0';
            end if;
        end if;
```

接下来，根据控制信号x的值，对输出信号c进行赋值。如果x为1，当QI达到"1111"时，c被赋值为'1'，否则为'0'。如果x为0，当QI达到"0000"时，c被赋值为'1'，否则为'0'。

最后，将QI转换为std_logic_vector类型并赋值给输出信号Q。

关键点拨

1. 当x='0'时，未对QI进行减1操作。
2. 当x='1'和x='0'时，都对c进行了赋值操作。
3. 当x='1'时，判断QI是否等于"1111"，当x='0'时，判断QI是否等于"0000"。

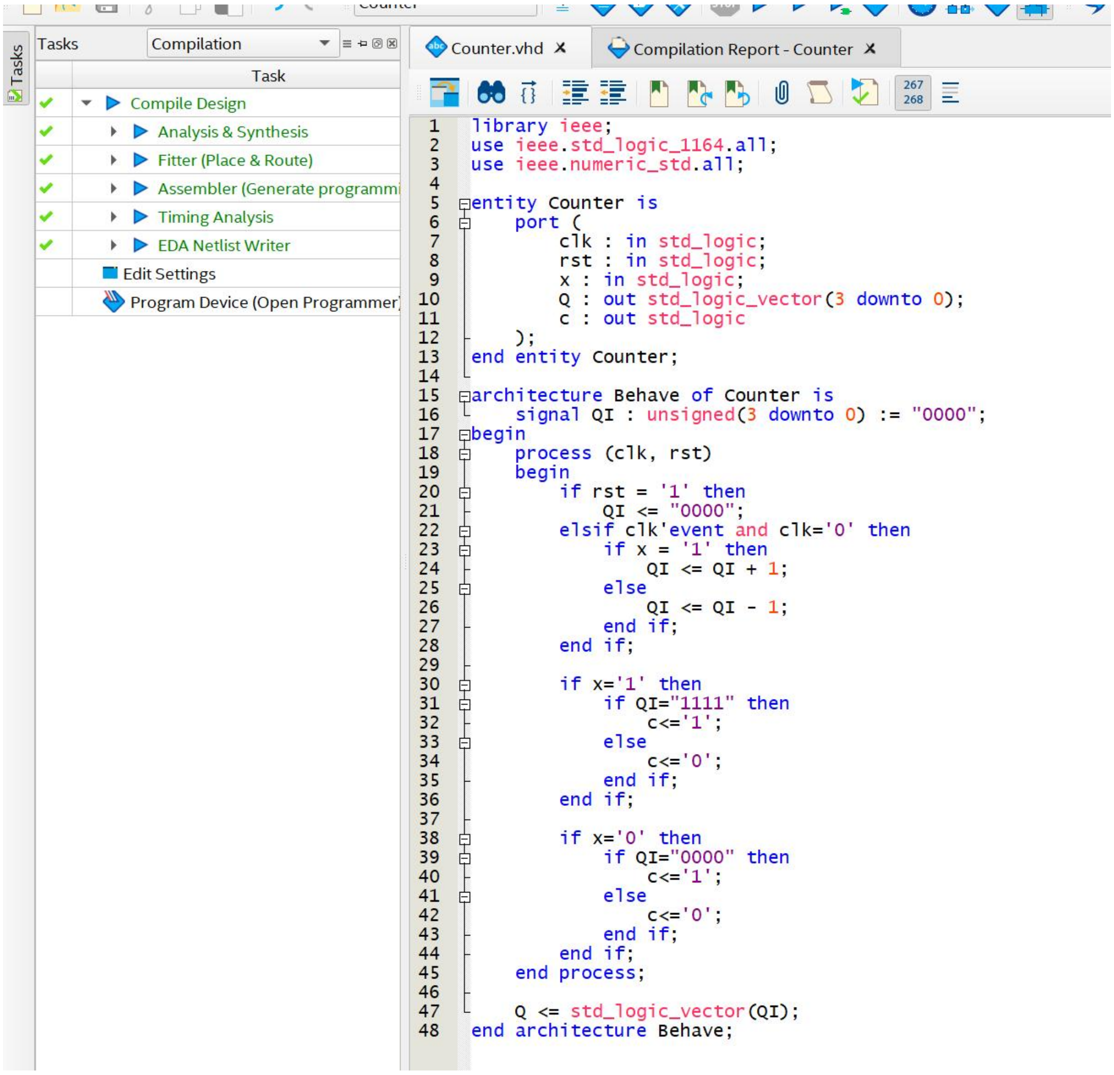
详细指导

1. 在代码中，当x='0'时，没有对QI进行减1操作。您应该添加一个else分支来处理这种情况。
2. 在代码中，当x='1'和x='0'时，都有对c进行赋值的操作。您应该将这两个if语句合并为一个，以减少重复代码。
3. 在代码中，当x='1'时，判断QI是否等于"1111"，而当x='0'时，判断QI是否等于"0000"。您应该将这两个判断条件合并为一个，以减少重复代码。

正确代码

[正确代码]

代码对比
你可以尝试继续和码上聊聊...
如需修改问题请点击左侧的重新提问按钮



Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programmi
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
Counter.vhd
Compilation Report - Counter
267
268

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Counter is
    port (
        clk : in std_logic;
        rst : in std_logic;
        x : in std_logic;
        Q : out std_logic_vector(3 downto 0);
        c : out std_logic
    );
end entity Counter;

architecture Behave of Counter is
    signal QI : unsigned(3 downto 0) := "0000";
begin
    process (clk, rst)
    begin
        if rst = '1' then
            QI <= "0000";
        elsif clk'event and clk='0' then
            if x = '1' then
                QI <= QI + 1;
            else
                QI <= QI - 1;
            end if;
        end if;

        if x='1' then
            if QI="1111" then
                c<='1';
            else
                c<='0';
            end if;
        end if;

        if x='0' then
            if QI="0000" then
                c<='1';
            else
                c<='0';
            end if;
        end if;
    end process;

    Q <= std_logic_vector(QI);
end architecture Behave;
```

编译成功。

代码如下：

library ieee;

```
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Counter is
        port (
                clk : in std_logic;
                rst : in std_logic;
                x : in std_logic;
                Q : out std_logic_vector(3 downto 0);
                c : out std_logic
        );
end entity Counter;

architecture Behave of Counter is
        signal QI : unsigned(3 downto 0) := "0000";
begin
        process (clk, rst)
        begin
                if rst = '1' then
                        QI <= "0000";
                elsif clk'event and clk='0' then
                        if x = '1' then
                                QI <= QI + 1;
                        else
                                QI <= QI - 1;
                        end if;
                end if;

                if x='1' then
                        if QI="1111" then
                                c<='1';
                        else
                                c<='0';
                        end if;
                end if;

                if x='0' then
                        if QI="0000" then
                                c<='1';
                        else
                                c<='0';
                        end if;
                end if;
        end process;
        Q <= std_logic_vector(QI);
end architecture Behave;
```

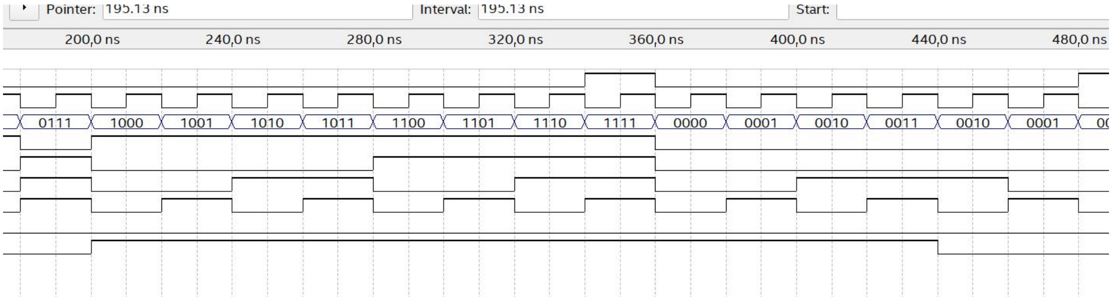
代码解释：在进程中，每当 clk 到达负边沿时进行一次计数，控制端 x 值为 1 时为加法计数，x 值为 0 时为减法计数。

若为加法计数，则 QI 到达 1111 时进位 c 输出 1，若为减法计数，则 QI 到达 0000 时输出 1，进程结束时将 QI 转化为逻辑向量 Q 输出。

rst=1 时计数器复位。

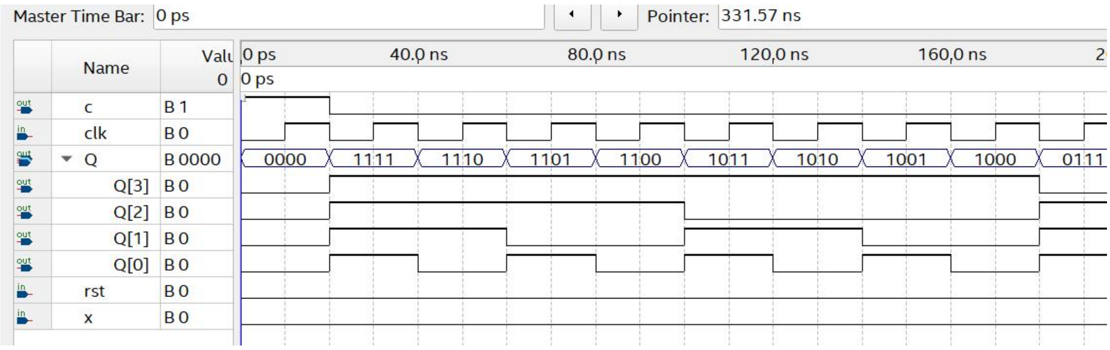
波形图：

加法（x=1）：



达到 1111 时产生进位，c 输出 1。

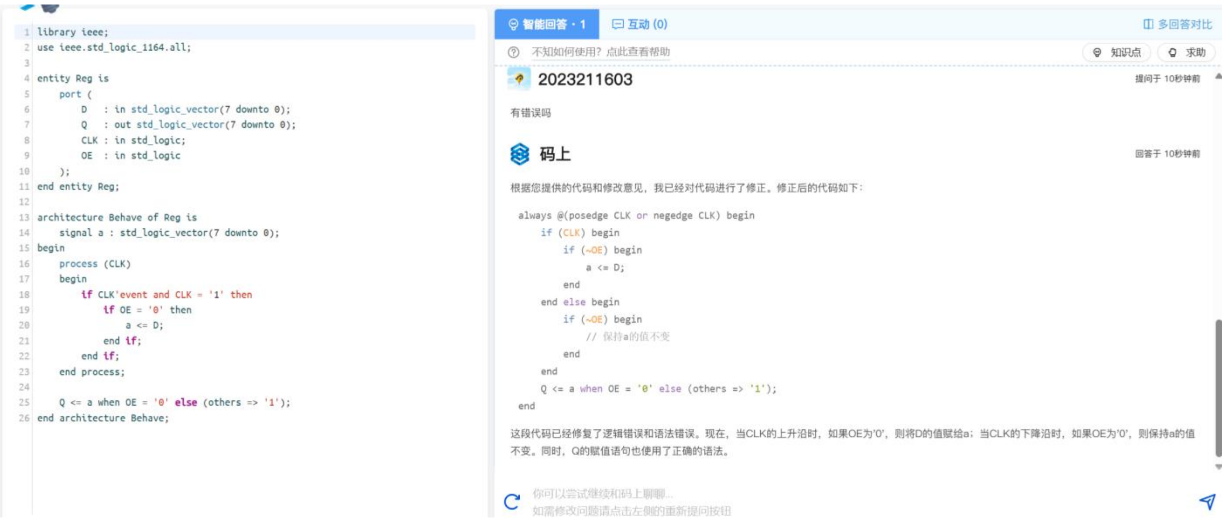
减法（x=0）：



达到 0000 时产生借位，c 输出 1。

实验三：8 位寄存器 74374

完成代码后上传“码上”进行纠错。


```
library ieee;
use ieee.std_logic_1164.all;

entity Reg is
    port (
        D   : in std_logic_vector(7 downto 0);
        Q   : out std_logic_vector(7 downto 0);
        CLK : in std_logic;
        OE  : in std_logic
    );
end entity Reg;

architecture Behave of Reg is
    signal a : std_logic_vector(7 downto 0);
begin
    process (CLK)
    begin
        if CLK'event and CLK = '1' then
            if OE = '0' then
                a <= D;
            end if;
        end if;
    end process;

    Q <= a when OE = '0' else (others => '1');
end architecture Behave;
```

2023211603 提问于 10秒钟前

有错误吗

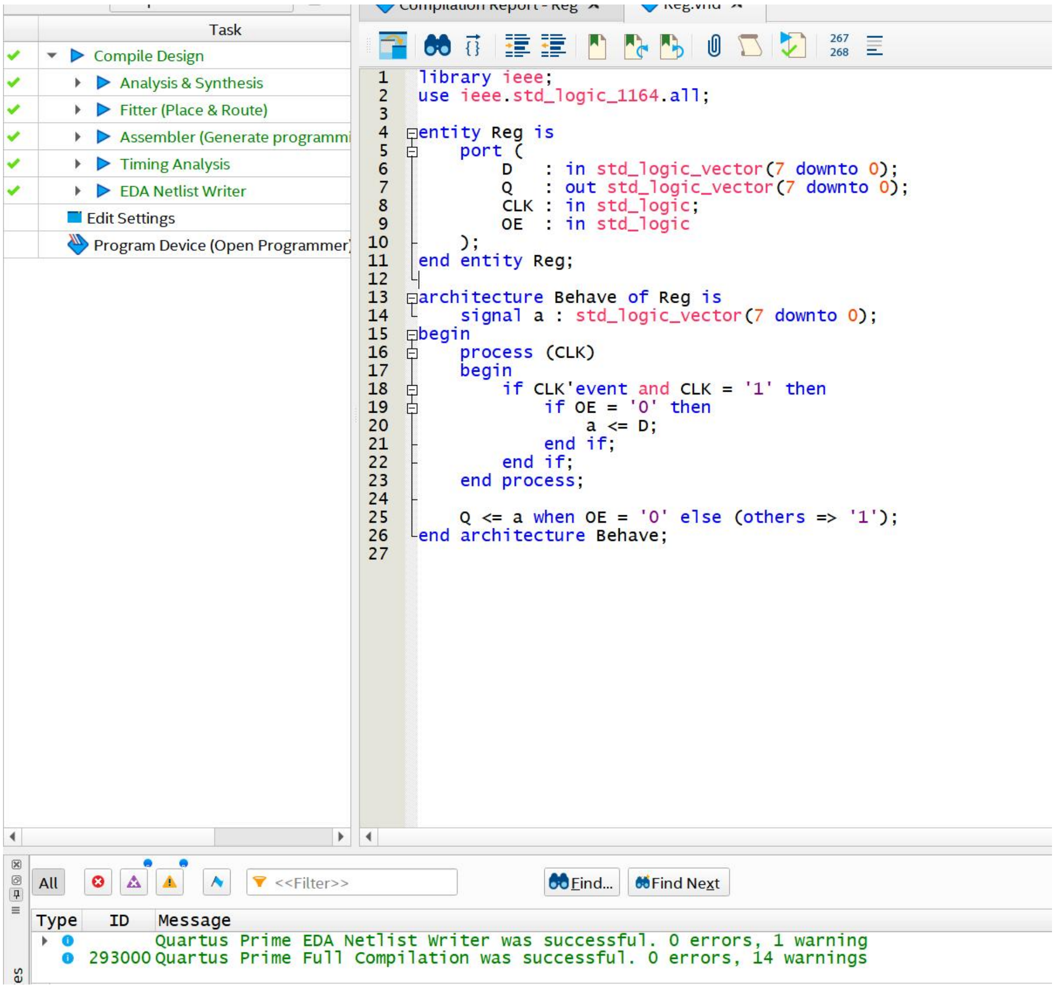
码上 回答于 10秒钟前

根据您提供的代码和修改意见，我已经对代码进行了修正。修正后的代码如下：

```
always @(posedge CLK or negedge CLK) begin
    if (CLK) begin
        if (~OE) begin
            a <= D;
        end
    end else begin
        if (~OE) begin
            // 保持a的值不变
        end
    end
    Q <= a when OE = '0' else (others => '1');
end
```

这段代码已经修复了逻辑错误和语法错误。现在，当CLK的上升沿时，如果OE为'0'，则将D的值赋给a；当CLK的下降沿时，如果OE为'0'，则保持a的值不变。同时，Q的赋值语句也使用了正确的语法。

你可以尝试继续和码上聊聊...
如需修改问题请点击左侧的重新提问按钮



```
library ieee;
use ieee.std_logic_1164.all;

entity Reg is
    port (
        D   : in std_logic_vector(7 downto 0);
        Q   : out std_logic_vector(7 downto 0);
        CLK : in std_logic;
        OE  : in std_logic
    );
end entity Reg;

architecture Behave of Reg is
    signal a : std_logic_vector(7 downto 0);
begin
    process (CLK)
    begin
        if CLK'event and CLK = '1' then
            if OE = '0' then
                a <= D;
            end if;
        end if;
    end process;

    Q <= a when OE = '0' else (others => '1');
end architecture Behave;
```

```
Type  ID     Message
            Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings
```

编译成功。

代码如下：

```
library ieee;
use ieee.std_logic_1164.all;

entity Reg is
	port (
            D       : in std_logic_vector(7 downto 0);
            Q       : out std_logic_vector(7 downto 0);
            CLK : in std_logic;
            OE     : in std_logic
        );
end entity Reg;

architecture Behave of Reg is
        signal a : std_logic_vector(7 downto 0);
begin
        process (CLK)
        begin
            if CLK'event and CLK = '1' then
                if OE = '0' then
                    a <= D;
                end if;
            end if;
        end process;

        Q <= a when OE = '0' else (others => '1');
end architecture Behave;
```

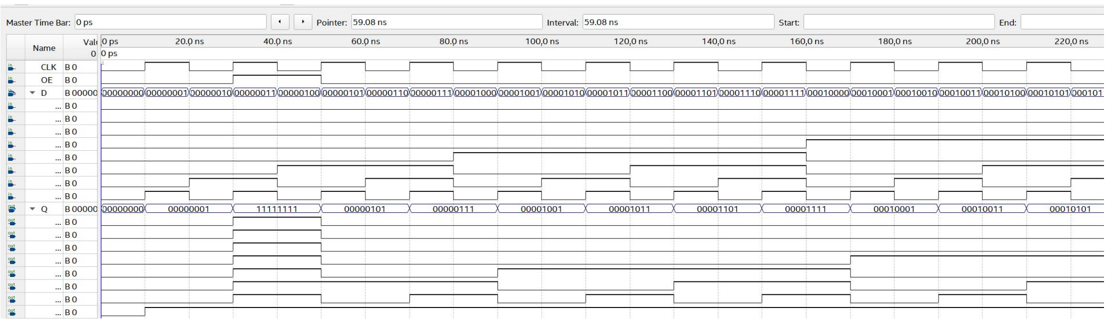
代码解释：

声明了一个内部信号 a，用作寄存器的内部状态，与输出 Q 宽度相同。如果输出使能信号 OE 为低电平，则在时钟上升沿将输入 D 的数据赋值给内部信号 a。

输出 Q 在 OE 为'0'时与内部信号 a 相同，这意味着寄存器的数据被输出。

如果 OE 为高电平，则输出 Q 被设置为高阻抗状态，输出 Q 保持 1。

波形图：



Master Time Bar: 0 ps
Pointer: 59.08 ns
Interval: 59.08 ns
Start:
End:

当 OE 为低电平时，寄存器正常工作，OE 为高电平时，Q 输出高电平。