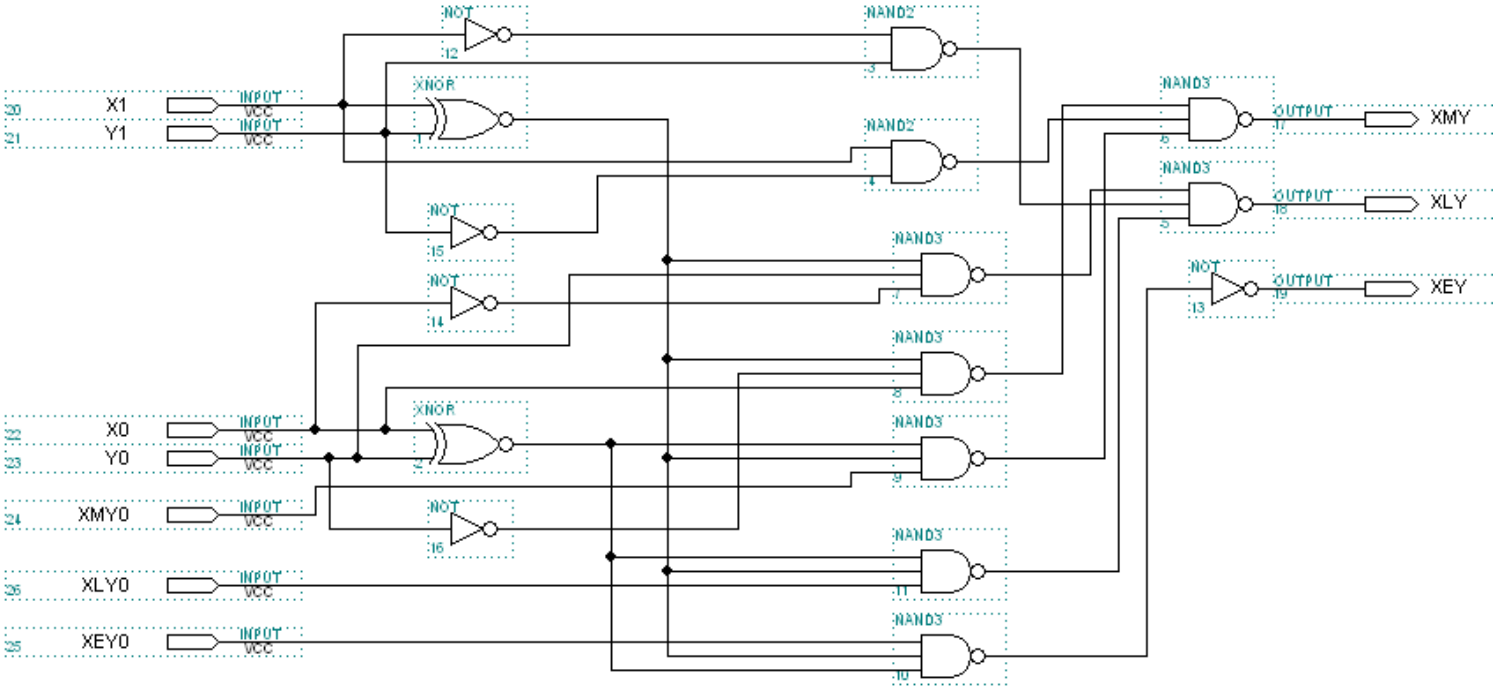
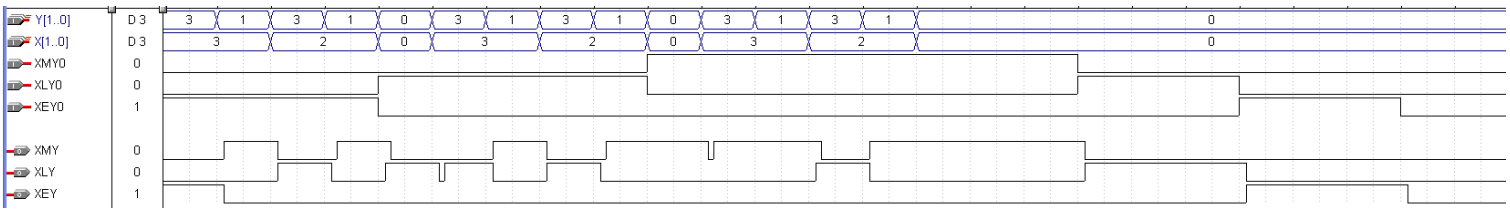
Пункт 1. Схема компаратора для двоичных слов из двух разрядов



Выходы:
XMY - X more Y (Больше)
XLY - X less Y (Меньше)
XEY - X equal Y (Равный)

Входы:
X(0,1); Y(0,1) - двоичные двухразрядные слова
XMY0, XLY0, XEY0 - необходимы для увеличения разрядности, то есть если сравниваются не двухбитные, а четырехбитные (что реализуем дальше), то на эти входы подаются сигналы с соответствующих выходов такого же двухбитного компаратора, который сравнивает два младших разряда.

Таблица истинности для п.1.

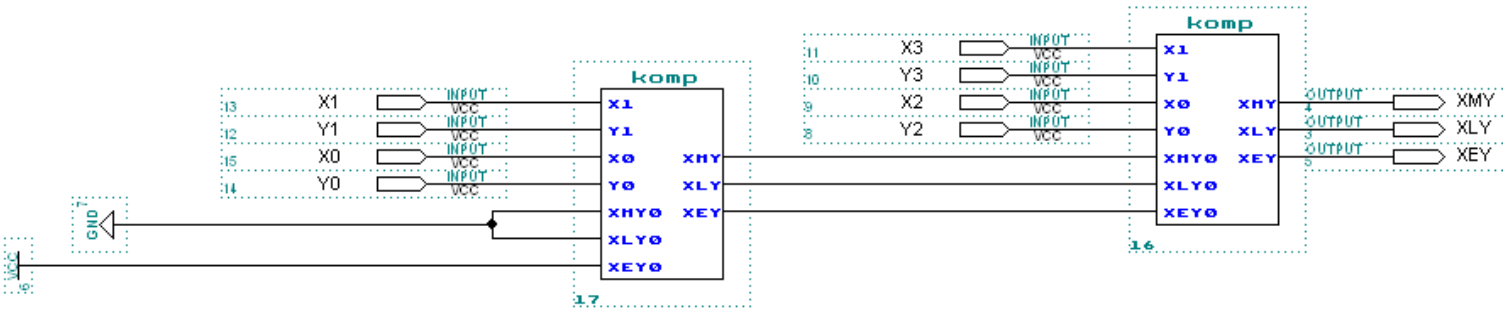


На выходах XMY0, XLY0, XEY0 мы симулируем, что 2 младших разряда X,Y больше, меньше или равны, то есть сравниваем уже 2 старших разряда с данными условиями.

Поведенческий VHD код п.1.

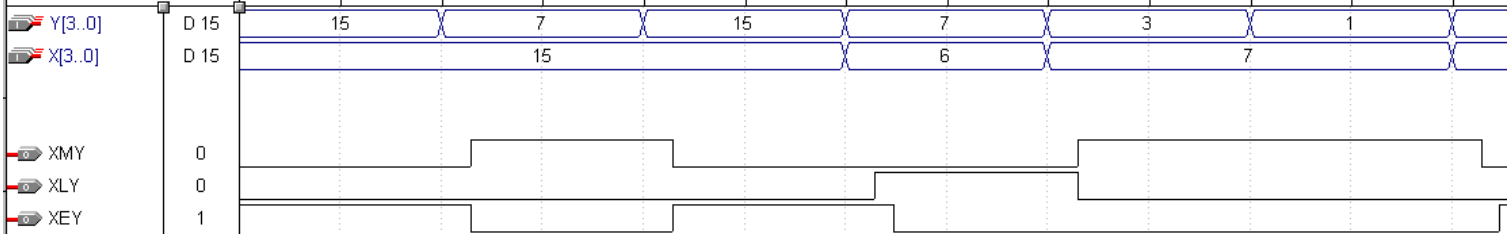
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY punkt4 IS
PORT (
        x: in std_logic_vector(1 downto 0);
        y: in std_logic_vector(1 downto 0);
        xmy0, xly0, xey0 : in std_logic;
        xmy, xly, xey : out std_logic
);
end punkt4;
Architecture behave of punkt4 is
begin
        Process(x,y,xmy0,xey0,xly0)
        begin
                        if(x>y) then
                                xly<='0';
                                xey<='0';
                                xmy<='1';
                        elsif(x<y) then
                                xly<='1';
                                xey<='0';
                                xmy<='0';
                        else
                                xmy<=xmy0;
                                xly<=xly0;
                                xey<=xey0;
                        end if;
        end process;
end behave;
```

Пункт 2. Схема компаратора с увеличенной вдвое разрядностью (сравниваются двоичные четырехбитные слова)



Тут мы на первом компараторе в XMY0, XLY0 подаем заземление '0' и симулируем, что 2 младших разряда (которых нет) равны, далее на 2 компаратор передаем значения с первого.

Таблица истинности для п.2.



Структурный VHD код п.2.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY punkt5 IS
PORT (x0: in std_logic_vector(1 downto 0);
        y0: in std_logic_vector(1 downto 0);
        x1: in std_logic_vector(1 downto 0);
        y1: in std_logic_vector(1 downto 0);
        xmy0, xly0, xey0 : in std_logic;
        xmy, xly, xey : out std_logic
);
end punkt5;

architecture behave of punkt5 is
        signal tmpxmy0, tmpxly0, tmpxey0: std_logic;
        component punkt4
                port(x: in std_logic_vector(1 downto 0);
                        y: in std_logic_vector(1 downto 0);
                        xmy0, xly0, xey0 : in std_logic;
                        xmy, xly, xey : out std_logic
                );
        end component;


        begin
        cmp1: punkt4
                port map(x0, y0, xmy0, xly0, xey0, tmpxmy0, tmpxly0, tmpxey0);
        cmp2: punkt4
                port map(x1, y1, tmpxmy0, tmpxly0, tmpxey0, xmy, xly, xey);

end behave;

configuration con of punkt5 is
        for behave
                for cmp1, cmp2: punkt4
                        use entity work.punkt4(behave);
                end for;
        end for;
end con;
```

*Симуляция в квартусе точно такая же, что и в максе с графическим представлением компаратов