VHDL

Multiple architectures for a single entity

- In VHDL you may have a single entity but you could write multiple architectures and at the time of synthesizing or compiling for simulation you can say which particular architecture you need to use.
- When you have multiple architectures, you have to choose which particular architecture you are going to use before you stimulate or synthesize at beginning. It has to be specified and that syntax is called configuration.

What is the point of specifying multiple architectures?

1. Simulation vs. Synthesis

- There are two main purposes of any HDL: Simulation and Synthesis of digital circuits. However, historically it all started for documentation (describing the circuit for documenting).
- In VHDL, there has anything you write you know syntactically correct, logically correct can be simulated but it may not be synthesizable.
- If you know the syntax of VHDL, you write any code you go for simulation (using simulator) and that whatever you have written can be simulated, but unless you take care it may not be synthesized.
- To design a complex digital system, initially, you may want to simply write a quick code to verify your idea (of designing that system) without much thinking about synthesis and then you can work on how to write a code which can be synthesized.

2. Satisfying design constraints ~ area and speed

- It is not that always possible to get a minimum area and maximum speed simultaneously. When you go for minimum area you might end up with large delay or less speed. Sometime when you give implement something with a large area (more logic elements) you might get more speed. (example: the ripple adder and carry look ahead adder).
- You might write some architecture which gives a very minimal area but less speed or which gives a really good speed but the area may not be minimal.

3. Target technology specific architecture

- You may write some code well-suited for FPGA/CPLD or you may define a architecture which will be optimal if you go for an ASIC.

The idea of providing multiple architectures (in VHDL) has a really good use.

VHDL modelling styles

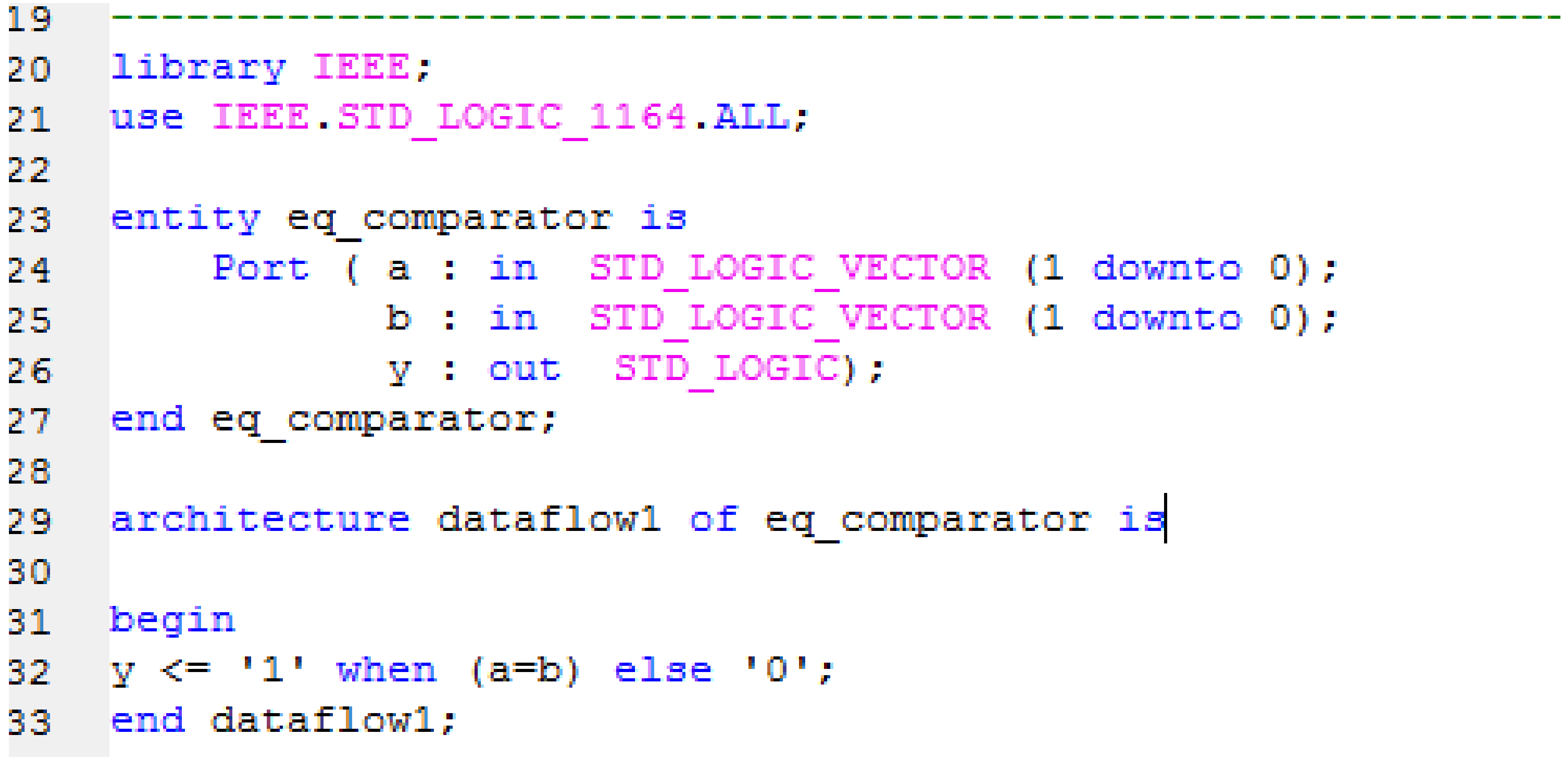
Various ways of describing the components of the circuit

- **Dataflow model (Concurrent statements)**

- **Behavioural model (Sequential statements with process + concurrent processes or statement which is concurrent with process)**

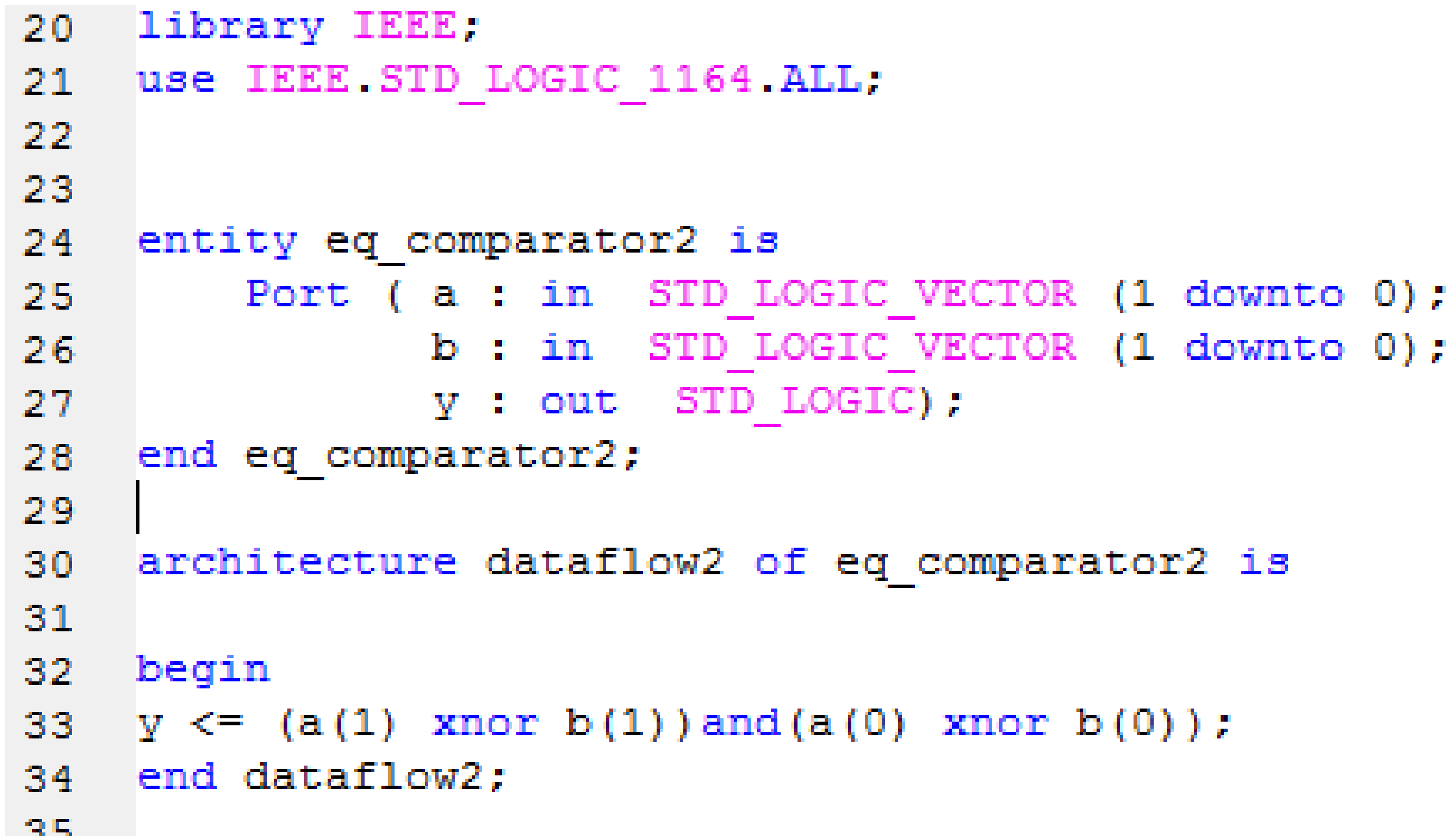
- **Structural model**

Dataflow model of 2-bit equality comparator



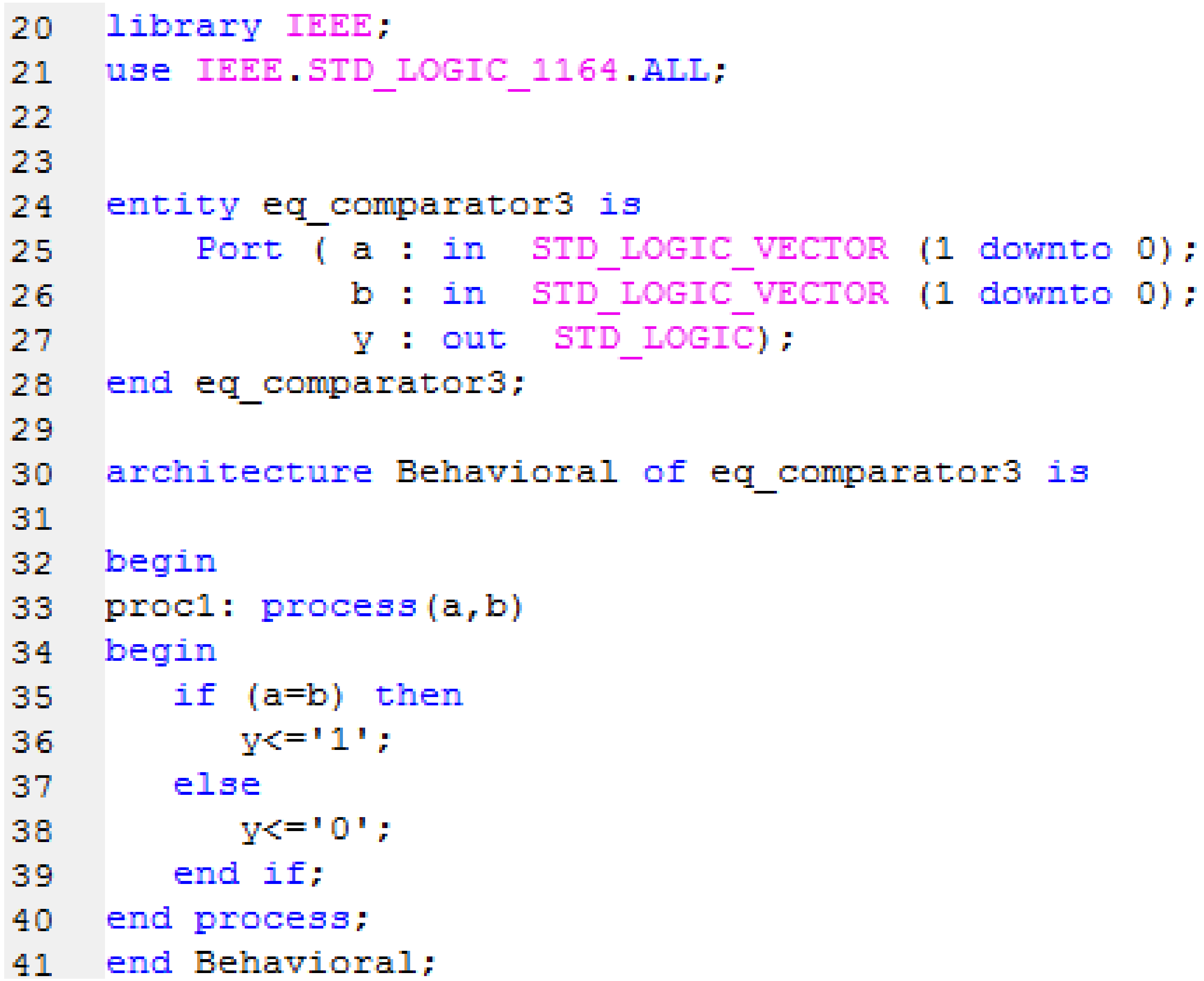
```
----------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity eq_comparator is
    Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
           b : in  STD_LOGIC_VECTOR (1 downto 0);
           y : out  STD_LOGIC);
end eq_comparator;

architecture dataflow1 of eq_comparator is

begin
y <= '1' when (a=b) else '0';
end dataflow1;
```

Another Dataflow model of 2-bit equality comparator



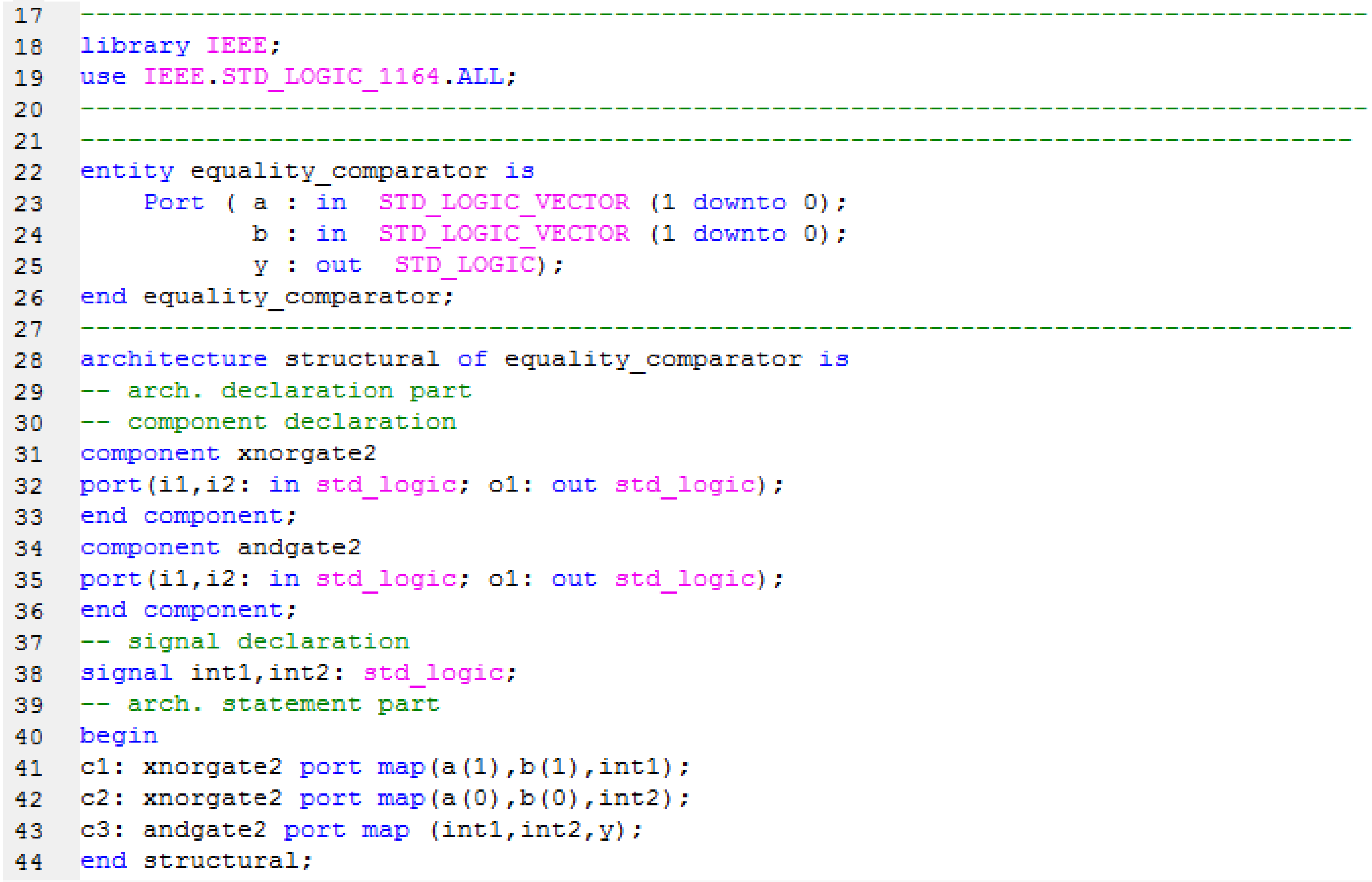
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity eq_comparator2 is
    Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
           b : in  STD_LOGIC_VECTOR (1 downto 0);
           y : out  STD_LOGIC);
end eq_comparator2;

architecture dataflow2 of eq_comparator2 is

begin
y <= (a(1) xnor b(1))and(a(0) xnor b(0));
end dataflow2;
```

Behavioral model of 2-bit equality comparator



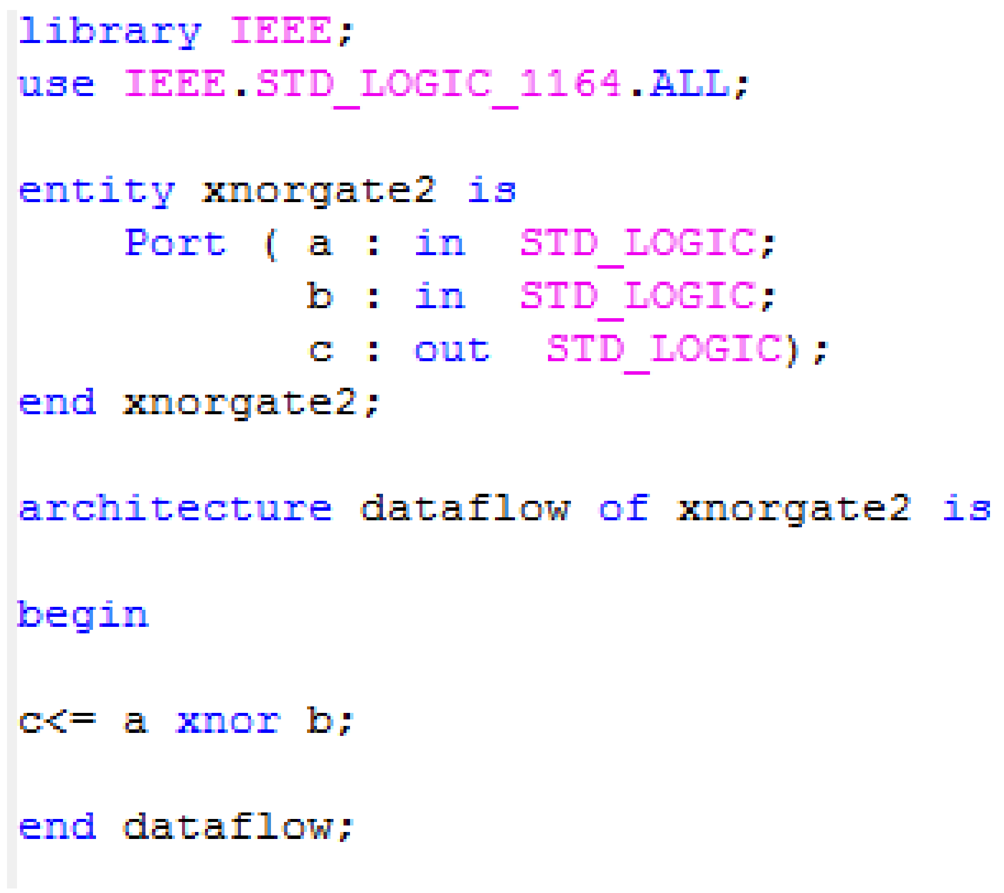
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity eq_comparator3 is
    Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
           b : in  STD_LOGIC_VECTOR (1 downto 0);
           y : out  STD_LOGIC);
end eq_comparator3;

architecture Behavioral of eq_comparator3 is

begin
proc1: process(a,b)
begin
   if (a=b) then
      y<='1';
   else
      y<='0';
   end if;
end process;
end Behavioral;
```

Structural codes

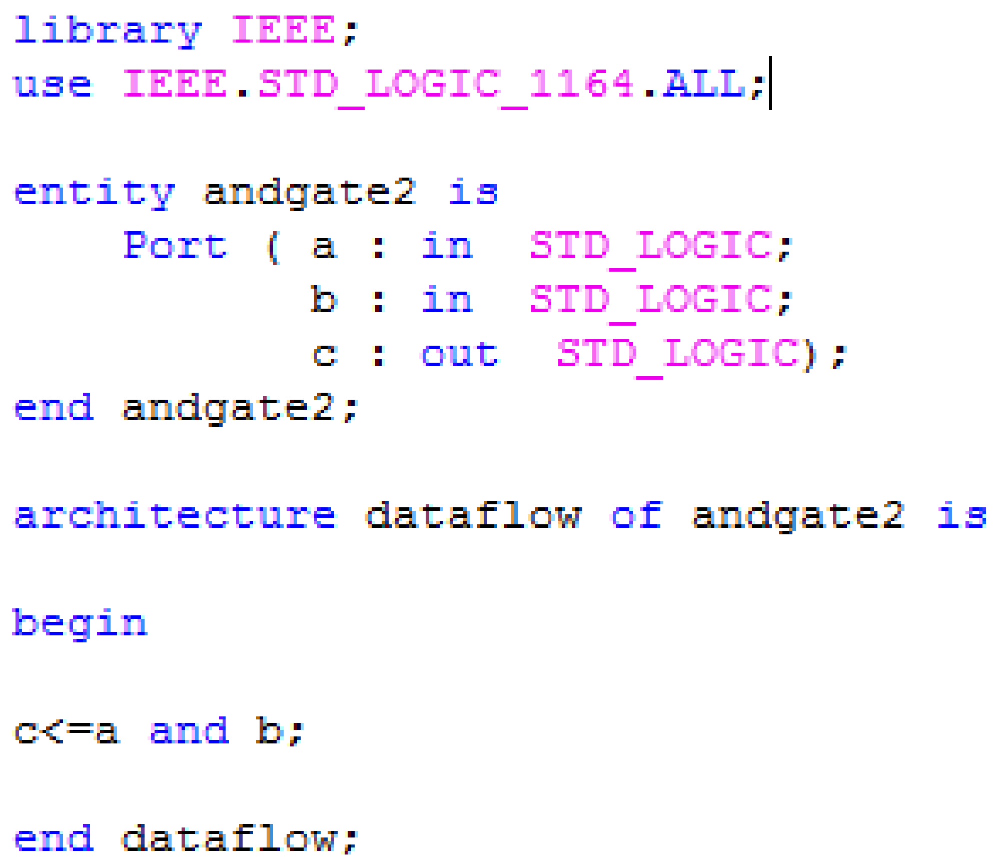


```vhdl
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
----------------------------------------------------------------------------------
---------------------------------------------------------------------------------
entity equality_comparator is
    Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
           b : in  STD_LOGIC_VECTOR (1 downto 0);
           y : out  STD_LOGIC);
end equality_comparator;
---------------------------------------------------------------------------------
architecture structural of equality_comparator is
-- arch. declaration part
-- component declaration
component xnorgate2
port(i1,i2: in std_logic; o1: out std_logic);
end component;
component andgate2
port(i1,i2: in std_logic; o1: out std_logic);
end component;
-- signal declaration
signal int1,int2: std_logic;
-- arch. statement part
begin
c1: xnorgate2 port map(a(1),b(1),int1);
c2: xnorgate2 port map(a(0),b(0),int2);
c3: andgate2 port map (int1,int2,y);
end structural;
```

Codes of Components



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

entity xnorgate2 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           c : out  STD_LOGIC);
end xnorgate2;

architecture dataflow of xnorgate2 is

begin

c<= a xnor b;

end dataflow;
```



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

entity andgate2 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           c : out  STD_LOGIC);
end andgate2;

architecture dataflow of andgate2 is

begin

c<=a and b;

end dataflow;
```