[5.2]
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY Excl_2 IS
  PORT ( 
    a: in std_logic;
    b: in std_logic;
    g: out std_logic
  );
END Excl_2;

ARCHITECTURE dataflow OF Excl_2 IS
BEGIN
  g <= a xor b;
END dataflow;

[5.3]
g <= (a and b) or (c and d);

[5.6]
\[ f = (a \cdot b) + (c \cdot (w \oplus z)) \]

[5.7]
\[ Out = (a + b) \cdot \overline{c} \cdot d \]

[5.8]
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY Prob5_8 IS
  PORT ( 
    a: in std_logic;
    b: in std_logic;
    c: in std_logic;
    F: out std_logic
  );
END Prob5_8;

ARCHITECTURE dataflow OF Prob5_8 IS
BEGIN
  F <= not( (not a) xor (not(b and c)));
END dataflow;
Here is another solution that uses signals in the architecture. Both of these architectures use dataflow designs.

```vhdl
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY Prob5_8 IS
  PORT ( a: in std_logic;
         b: in std_logic;
         c: in std_logic;
         F: out std_logic);
END Prob5_8;

ARCHITECTURE dataflow OF Prob5_8 IS
  SIGNAL w1, w2 : std_logic;
  BEGIN
    w1 <= not a;
    w2 <= not (b and c);
    F <= not (w1 xor w2);
  END dataflow;
```

```vhdl
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY Logic_box IS
  PORT ( a: in std_logic_vector(3 downto 0);
         b: in std_logic_vector(3 downto 0);
         Out_1: out std_logic;
         Out_2: out std_logic);
END Logic_box;

ARCHITECTURE dataflow OF Logic_box IS
  BEGIN
    Out_1 <= (a(3) xor b(3)) and (a(2) and b(2));
    Out_2 <= (a(1) or b(1)) or (a(0) and b(0));
  END dataflow;
```