Белорусский государственный университет информатики и радиоэлектроники

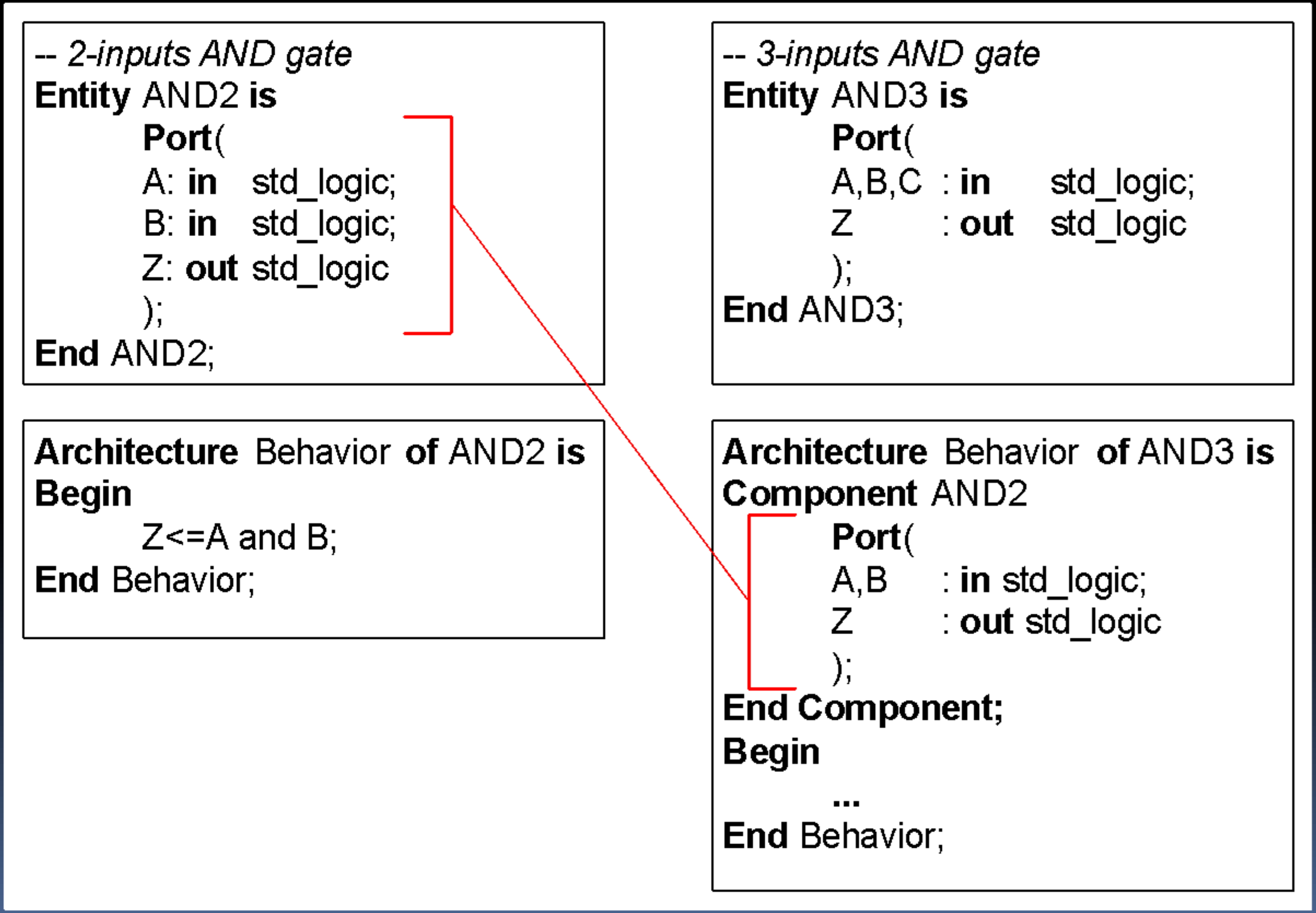
4-ый курс специальности ПОИТ

# ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЦИФРОВОГО ПРОЕКТИРОВАНИЯ

2020

## Introducing to VHDL

### Structural Descriptions:



```
-- 2-inputs AND gate
Entity AND2 is
        Port(
        A: in   std_logic;
        B: in   std_logic;
        Z: out std_logic
        );
End AND2;
```

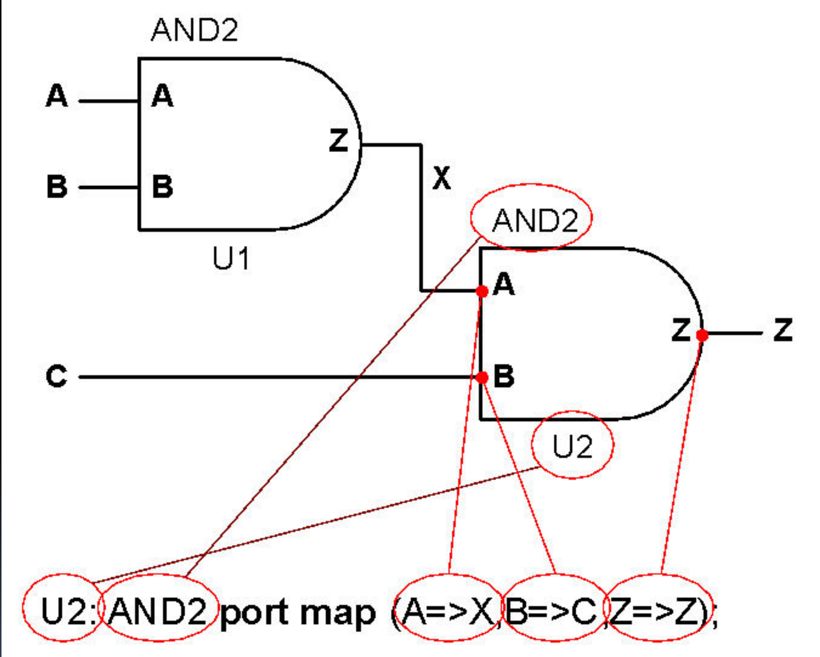
```
Architecture Behavior of AND2 is
Begin
        Z<=A and B;
End Behavior;
```

```
-- 3-inputs AND gate
Entity AND3 is
        Port(
        A,B,C  : in     std_logic;
        Z      : out    std_logic
        );
End AND3;
```

```
Architecture Behavior of AND3 is
Component AND2
        Port(
        A,B    : in std_logic;
        Z      : out std_logic
        );
End Component;
Begin
        ...
End Behavior;
```

## Introducing to VHDL

### Structural Descriptions:

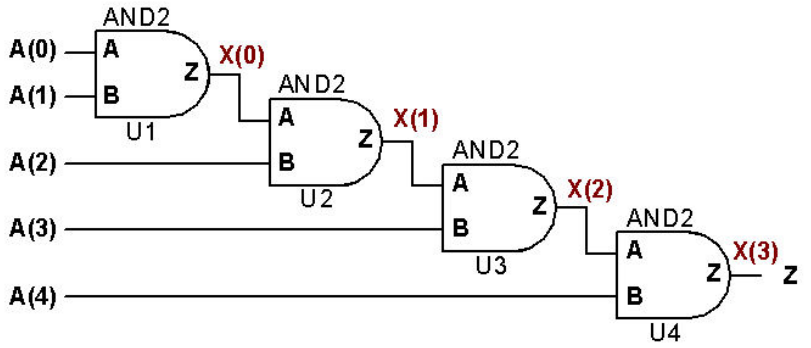


```
Architecture Behavior of AND3 is
Component AND2
        Port (
        A,B    : in std_logic;
        Z      : out std_logic
        );
signal X: std_logic;
End Component;
Begin
        -- named port mapping
        U1: AND2 port map (A=>A, B=>B, Z=>X);
        U2: AND2 port map (A=>X,B=>C,Z=>Z);

        -- unnamed port mapping
--      U1: AND2 port map (A, B, X);
--      U2: AND2 port map (X,C,Z);
End Behavior;
```

# Introducing to VHDL

## Structural Descriptions:



```
-- 5-inputs AND gate
Entity AND5 is
        Port(
        A       : in    std_logic_vector(4 downto 0);
        Z       : out   std_logic
        );
End AND5;
```

```
Architecture Behavior of AND5 is
Component AND2
        Port(
        A,B     : in std_logic;
        Z       : out std_logic
        );
End Component;
signal X: std_logic_vector(3 downto 0);
Begin
        GEN_0: AND2 port map (A(0),A(1),X(0));
        SCH: FOR J in 1 to 3 GENERATE
                GEN_J: AND2 port map (X(J-1),A(J+1),X(J));
        End GENERATE;
        Z<=X(3);
End Behavior;
```

# Introducing to VHDL

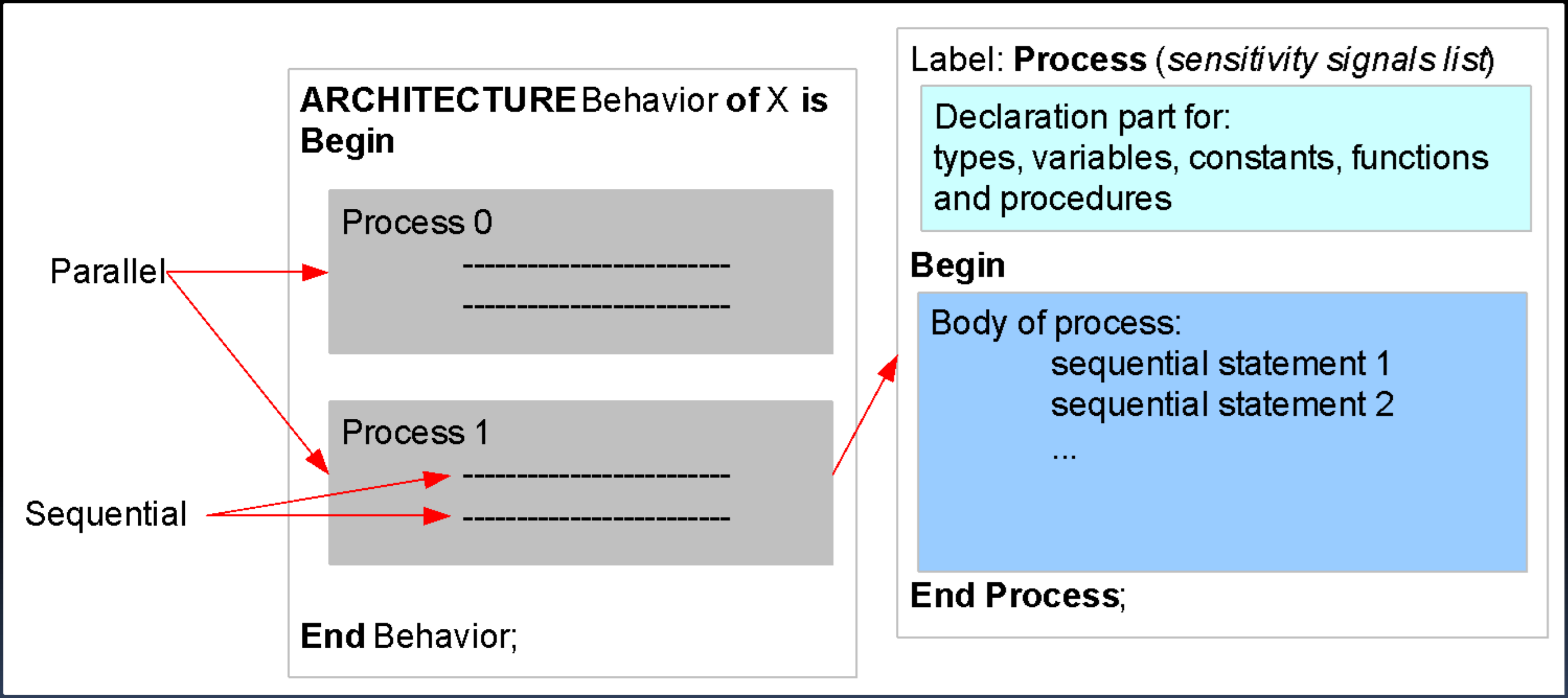
## Structural Descriptions:

```
-- N-inputs AND gate
Entity ANDN is
        Generic (N: integer :=10);
        Port(
        A       : in    std_logic_vector( N-1 downto 0);
        Z       : out   std_logic
        );
End ANDN;
```

```
Architecture Behavior of ANDN is
Component AND2
        Port(
        A,B     : in std_logic;
        Z       : out std_logic
        );
End Component;
signal X: std_logic_vector( N-2 downto 0);
Begin
        GEN_0: AND2 port map (A(0),A(1),X(0));
        SCH: FOR J in 1 to N-2 GENERATE
                GEN_J: AND2 port map (X(J-1),A(J+1),X(J));
        End GENERATE;
        Z<=X(N-2);
End Behavior;
```

# Introducing to VHDL

## Behavioral Descriptions (processes):


ARCHITECTURE Behavior of X is
Begin
Process 0
Process 1
End Behavior;
Parallel
Sequential
Label: Process (sensitivity signals list)
Declaration part for: types, variables, constants, functions and procedures
Begin
Body of process: sequential statement 1 sequential statement 2 ...
End Process;

# Introducing to VHDL

## Behavioral Descriptions (processes):

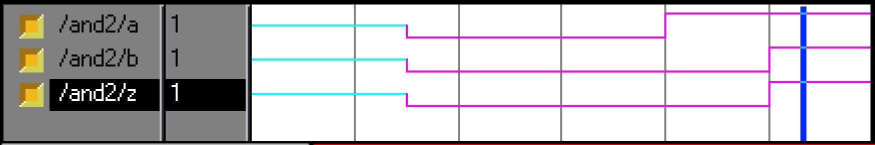
Sensitivity list: each process can be it two states:

1. Running (executing sequential statements)
2. Suspending (waiting for signals changing) .

At the beginning process is suspended. If one of the signals from sensitivity list was changed process begins to execute all sequential statements inside its body. After the last statement was executed process returns to the suspended state.

```
ENTITY AND2 IS
   port(
        A,B:  in std_logic;
        Z:  out std_logic
   );
END AND2;

--
ARCHITECTURE Behavior OF AND2 IS
BEGIN
   Main: process(A,B)
   begin
     Z<=A and B;
   end process;
END Behavior;
```



```
Main: process(A)
begin
  Z<=A and B;
end process;
```

**Error! Signal B is not in sensitivity list!**

```
Main: process
begin
  Z<=A and B;
end process;
```

**Infinitive loop! Process has no wait statement**

# Introducing to VHDL

## Behavioral Descriptions (processes):

```
ENTITY AND3 IS
   port(
        A,B,C:  in std_logic;
        Z:  out std_logic
   );
END AND3;

--
ARCHITECTURE Behavior OF AND3 IS
BEGIN
   Main:  process(A,B,C)
   variable X: std_logic;
    begin
           X:=A and B;
           Z<=X and C;
    end process;
END Behavior;
```

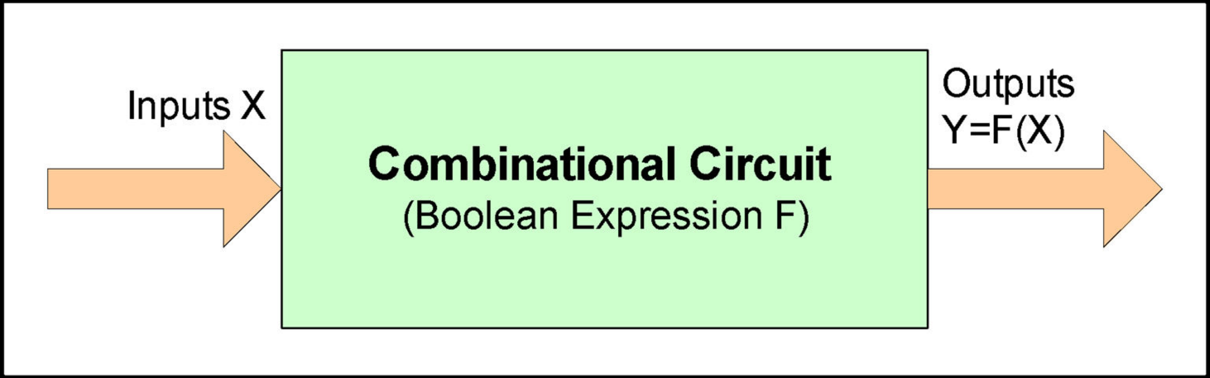
At this example two statements inside process will be sequentially simulated.

1. Process Main is in suspended state.
2. One of the signals A,B or C was changed.
3. Process recognized this change by sensitivity list and starts to execute sequential statements.
4. At first the value of variable X will be calculated X:=A and B.
5. Process stops until next changing of signals and Z will get a new value Z<=X and C;.

The simulation of these steps will be done in Zero simulation time!

## Introducing to VHDL

### Combinational Logic Description:



Combinational circuits can be described as behavioral and structural VHDL descriptions:

- Truth Table;
- Logic (Boolean) expression;
- Net of basic logic elements;
- High level operators and statements;

## Introducing to VHDL

### Combinational Logic Description:

Отрицание

| $a$ | $f(a) = \overline{a}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Конъюнкция

| $a$ | $b$ | $f(a,b) = a*b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

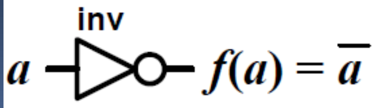
Дизъюнкция

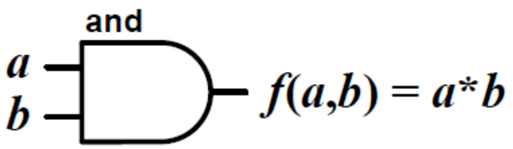
| $a$ | $b$ | $f(a,b) = a+b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

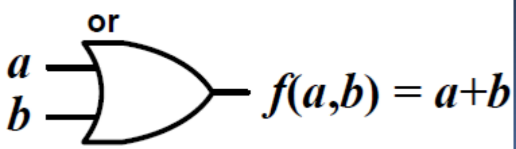
Инвертор (NOT)

Двухходовой вентиль И (AND)

Двухходовой вентиль ИЛИ (OR)







## Introducing to VHDL

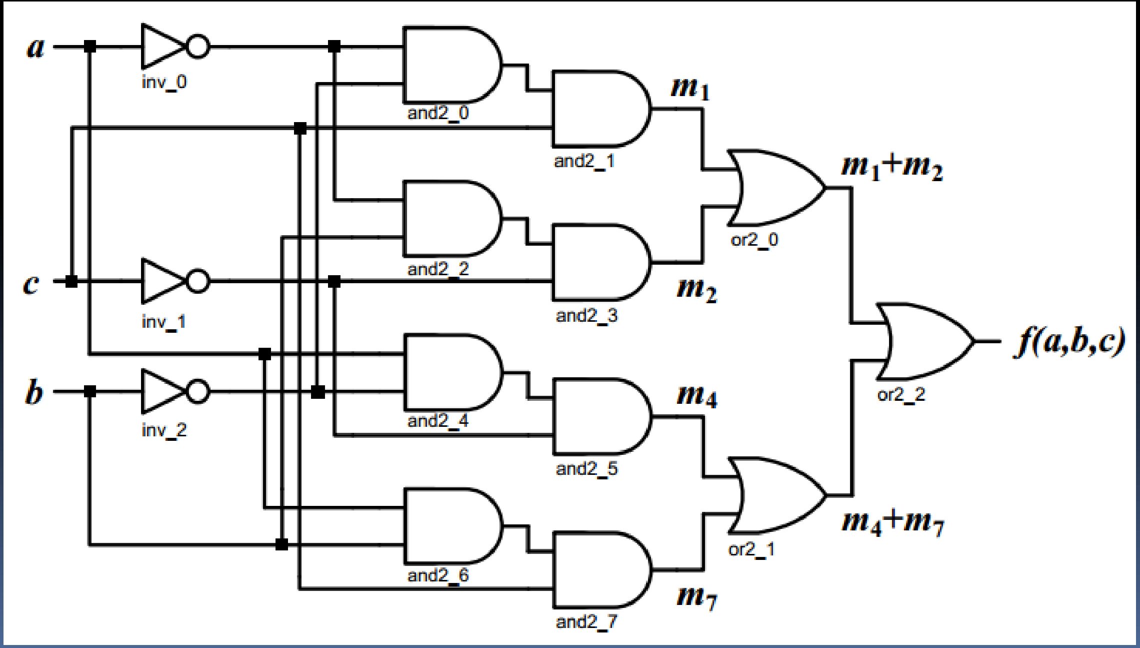
### Combinational Logic Description:

Таблица истинности
и минтермы функции $f(a,b,c)$

| a | b | c | Минтермы | $f(a,b,c)$ |
|---|---|---|---|---|
| 0 | 0 | 0 | $m_0 = \bar{a}\cdot\bar{b}\cdot\bar{c}$ | $f_0 = 0$ |
| 0 | 0 | 1 | $m_1 = \bar{a}\cdot\bar{b}\cdot c$ | $f_1 = 1$ |
| 0 | 1 | 0 | $m_2 = \bar{a}\cdot b\cdot\bar{c}$ | $f_2 = 1$ |
| 0 | 1 | 1 | $m_3 = \bar{a}\cdot b\cdot c$ | $f_3 = 0$ |
| 1 | 0 | 0 | $m_4 = a\cdot\bar{b}\cdot\bar{c}$ | $f_4 = 1$ |
| 1 | 0 | 1 | $m_5 = a\cdot\bar{b}\cdot c$ | $f_5 = 0$ |
| 1 | 1 | 0 | $m_6 = a\cdot b\cdot\bar{c}$ | $f_6 = 0$ |
| 1 | 1 | 1 | $m_7 = a\cdot b\cdot c$ | $f_7 = 1$ |

$$f(a,b,c) = f_1\cdot m_1 + f_2\cdot m_2 + f_4\cdot m_4 + f_7\cdot\ m_7.$$

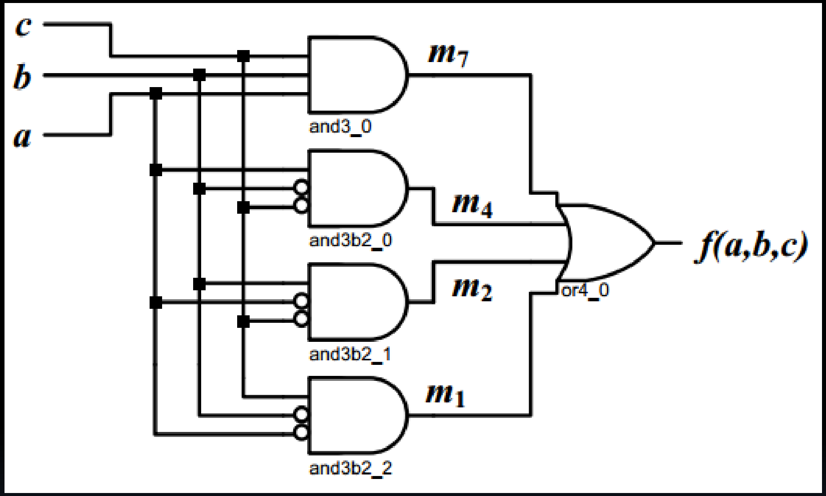
$$f(a,b,c) = \bar{a}\cdot\bar{b}\cdot c\ \ + \bar{a}\cdot b\cdot\bar{c} + a\cdot\bar{b}\cdot\bar{c}\ \ + a\cdot b\cdot c.$$



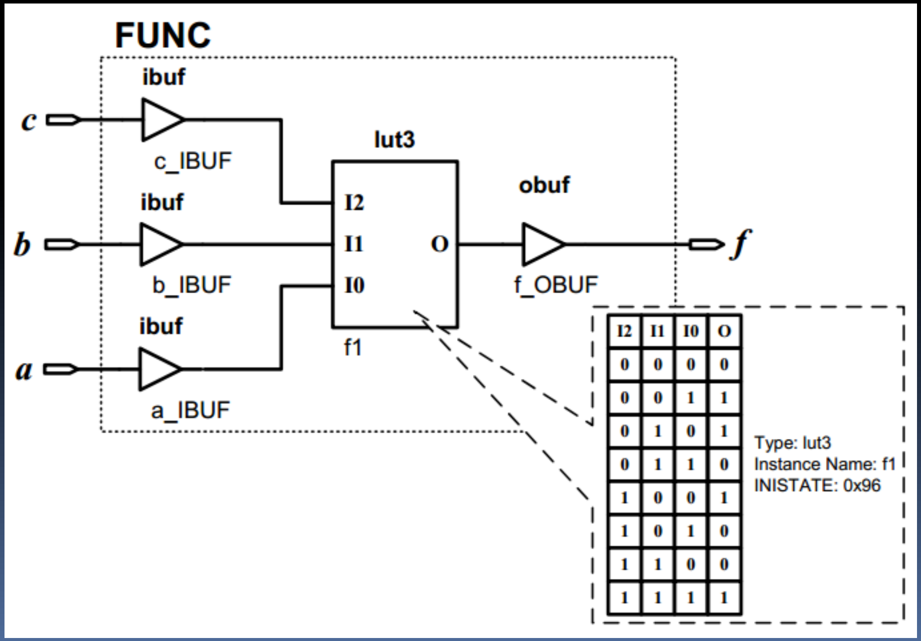
## Introducing to VHDL

### Combinational Logic Description:

Result of RTL-synthesis



Result of FPGA implementation



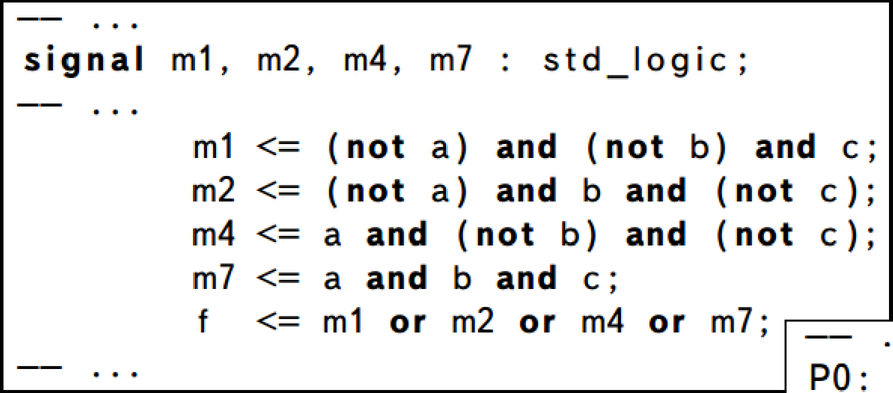
| I2 | I1 | I0 | O |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Type: lut3
Instance Name: f1
INISTATE: 0x96

## Introducing to VHDL

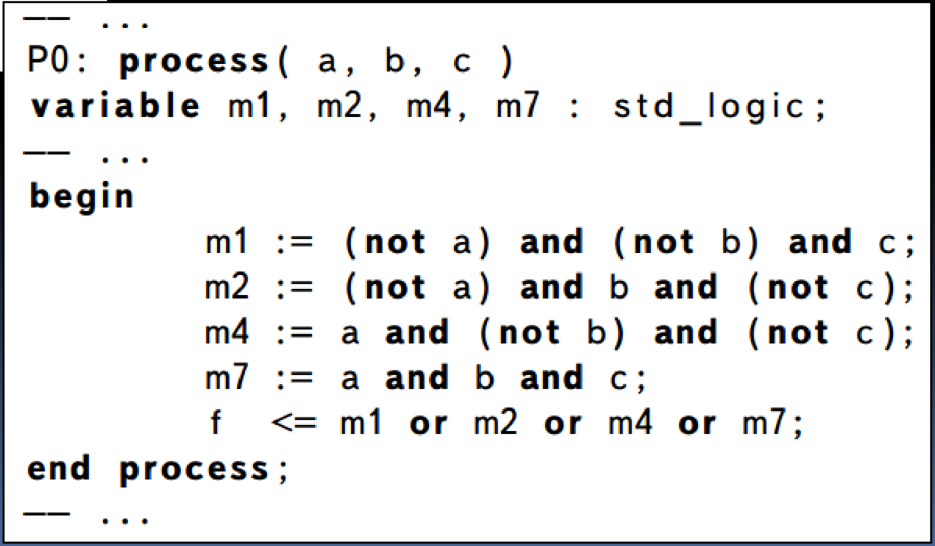
### Combinational Logic Description:

VHDL description #1



```
-- ...
signal m1, m2, m4, m7 : std_logic;
-- ...
        m1 <= (not a) and (not b) and c;
        m2 <= (not a) and b and (not c);
        m4 <= a and (not b) and (not c);
        m7 <= a and b and c;
        f  <= m1 or m2 or m4 or m7;
-- ...
```

VHDL description #2



```
-- ...
P0: process( a, b, c )
variable m1, m2, m4, m7 : std_logic;
-- ...
begin
        m1 := (not a) and (not b) and c;
        m2 := (not a) and b and (not c);
        m4 := a and (not b) and (not c);
        m7 := a and b and c;
        f  <= m1 or m2 or m4 or m7;
end process;
-- ...
```

## Introducing to VHDL

### Combinational Logic Description:

VHDL description #3 :-)

```
-- ...
        f <= a xor b xor c;
-- ...
```

VHDL description #4 with-select parallel operator

```
-- ...
signal abc: std_logic_vector(2 downto 0);
-- ...
        abc <= a & b & c;
        with abc select
        f <= '1' when "001" | "010" | "100" | "111",
             '0' when others;

-- ...
```

## Introducing to VHDL

### Combinational Logic Description:

VHDL description #5 case sequential operator

```
-- ...
P0: process ( a, b, c )
variable abc: std_logic_vector(2 downto 0);
-- ...
begin
        abc := a & b & c;
        case abc is
                when "001" | "010" | "100" | "111" =>
                                f <= '1';
                when others =>  f <= '0';
        end case;
end process;
-- ...
```

Combinational Circuit Description needs complete logic statements (all variants of assigned values)

## Introducing to VHDL

### Combinational Logic Description:

$$f(a,b,s) = a \cdot \overline{s} + b \cdot s$$

```
-- ...
P0: process ( a, b, s )
begin
        if s = '0' then
                f <= a;
        else
                f <= b;
        end if;
end process;
-- ...
```

?

```
-- ...
P0: process ( a, b, s )
begin
        if s = '0' then
                f <= a;
        else
                f <= 'Z';
        end if;
end process;
-- ...
```

?

```
-- ...
P0: process ( a, b, s )
begin
        if s = '0' then
                f <= a;
        end if;
end process;
-- ...
```

?

## Introducing to VHDL

### Combinational Logic Description:

$$f(a,b,c,s(1),s(0)) = a \cdot \overline{s(1)} \cdot \overline{s(0)} + \ b \cdot \overline{s(1)} \cdot s(0) + c \cdot s(1) \cdot \overline{s(0)}.$$

```
-- ...
P0: process ( a, b, c, s )
begin
        if s="00" then
                f <= a;
        elsif s="01" then
                f <= b;
        elsif s="10" then
                f <= c;
        end if;
end process;
-- ...
```

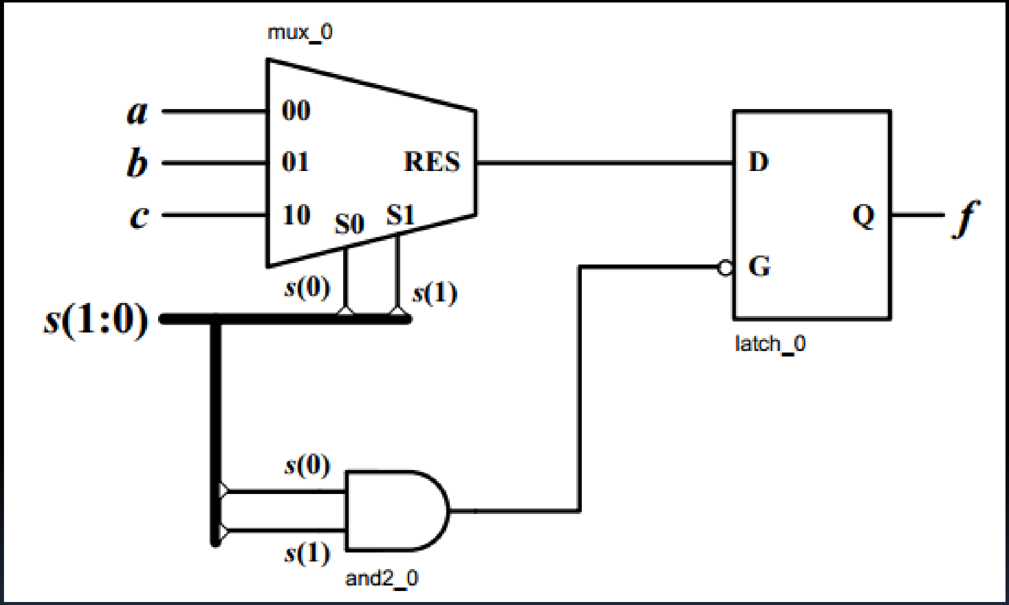
Where is a mistake?
a,b,c,s,f – are signals (ports)

## Introducing to VHDL

### Combinational Logic Description:

```
-- ...
P0: process ( a, b, c, s )
begin
        if s="00" then
                f <= a;
        elsif s="01" then
                f <= b;
        elsif s="10" then
                f <= c;
        end if;
end process;
-- ...
```

Behavioral Description



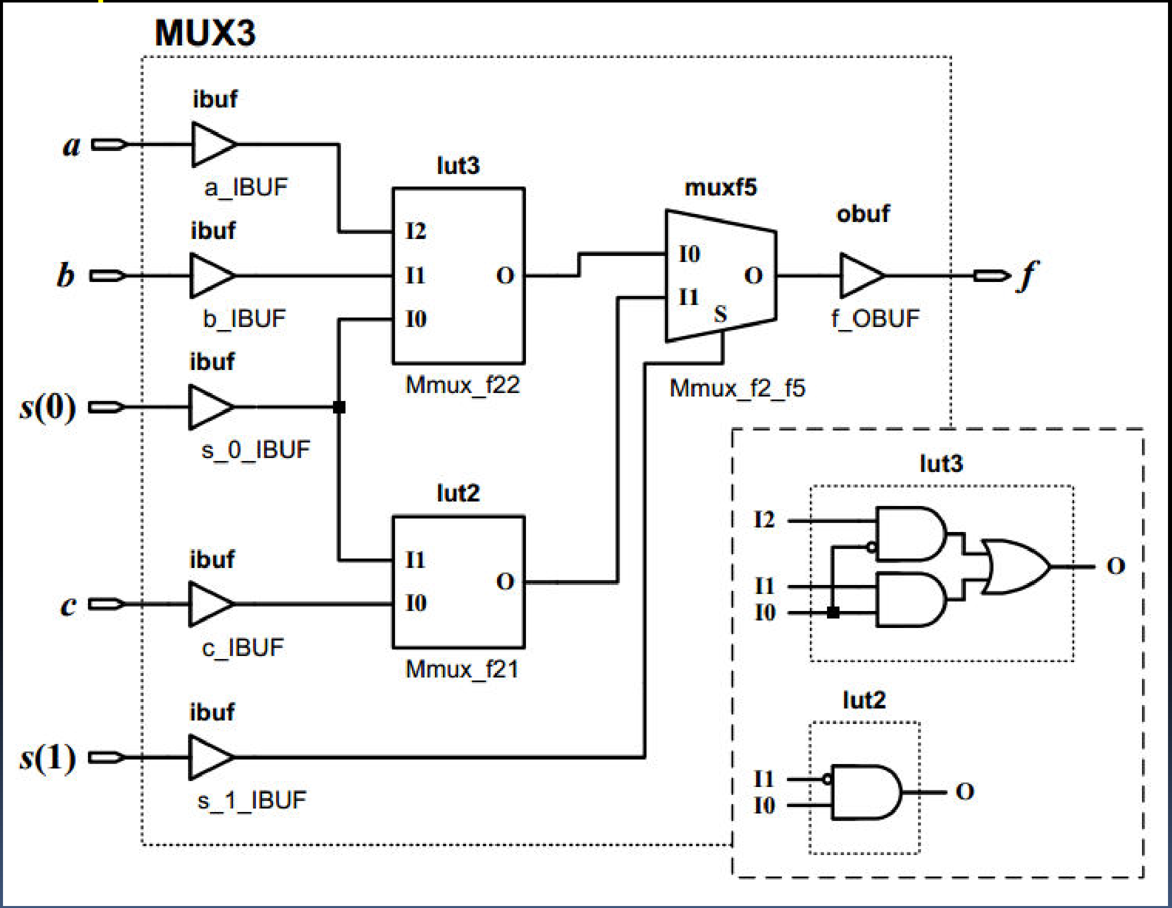
Result of RTL-synthesis

# Introducing to VHDL

## Combinational Logic Description:

```
 -- ...
 P0: process ( a, b, c, s )
 begin
         if s="00" then
                 f <= a;
         elsif s="01" then
                 f <= b;
         elsif s="10" then
                 f <= c;
         else
                 f <= '0';
         end if;
 end process;
 -- ...
```

Good Behavioral Description



Result of Schematic Synthesis