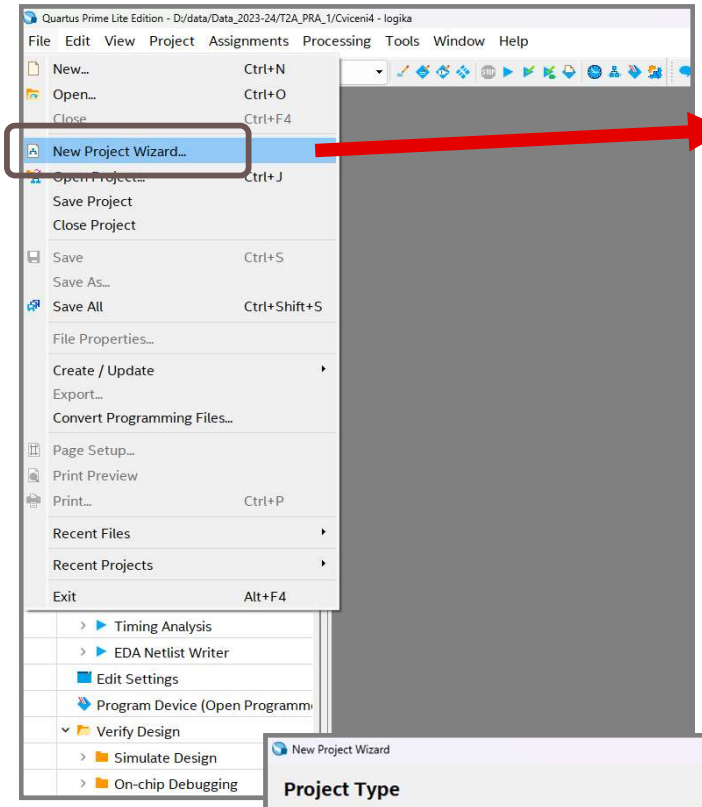
Quartus

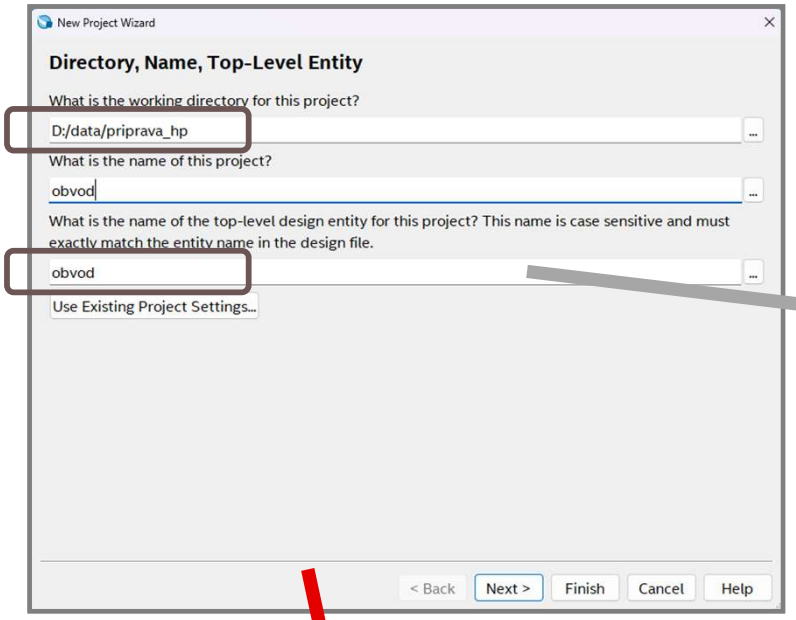
Vytvoření aplikace

Postup vytvoření aplikace

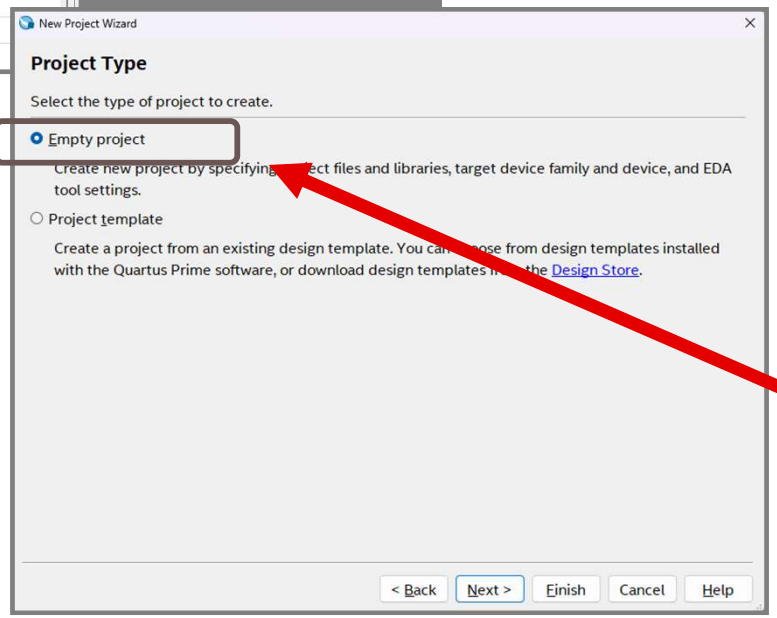
1. Umístění projektu, volba typu HW
2. Volba typu aplikace (schéma nebo VHDL)
3. Kontrola syntaxe
4. Připojení markerů k vývodům FPGA
5. Generování programovacího souboru
6. Nahrání do přípravku

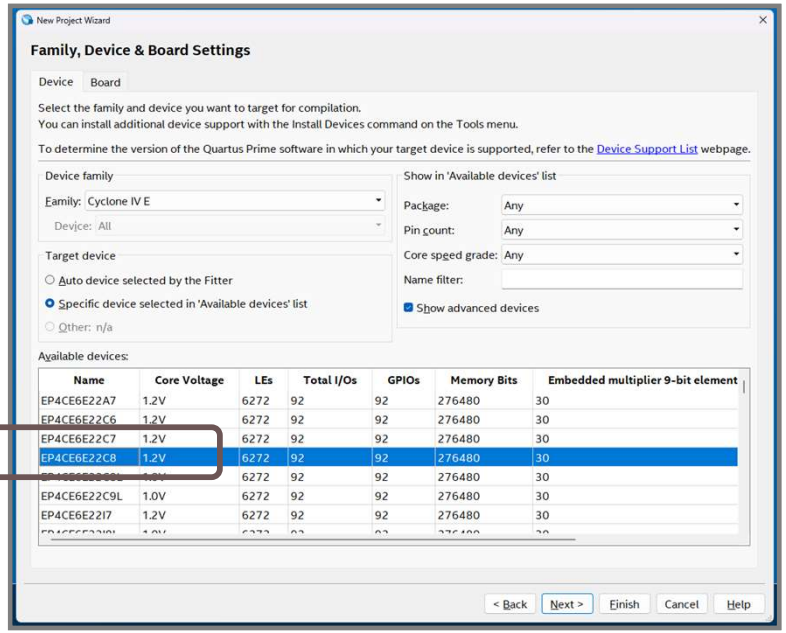
1. Umístění projektu a volba HW





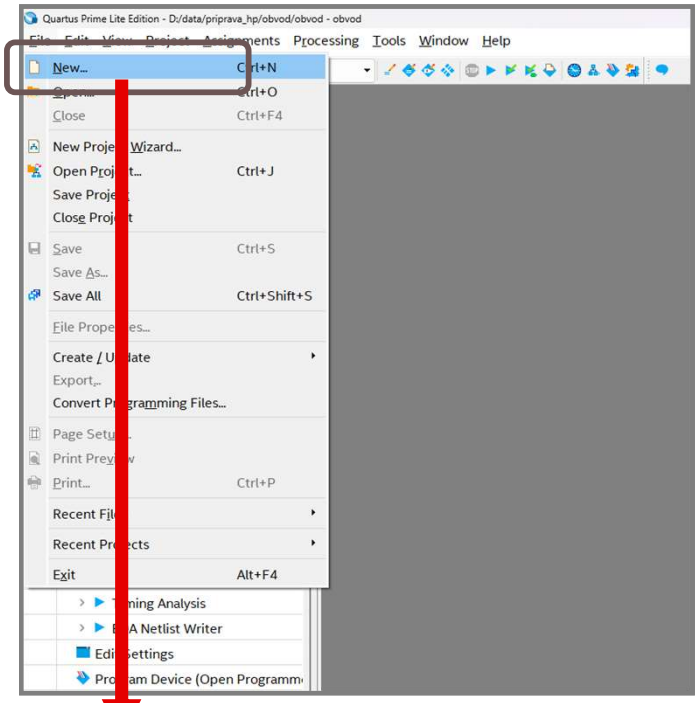
Top level designe entity
Nastavení v souboru VHDL

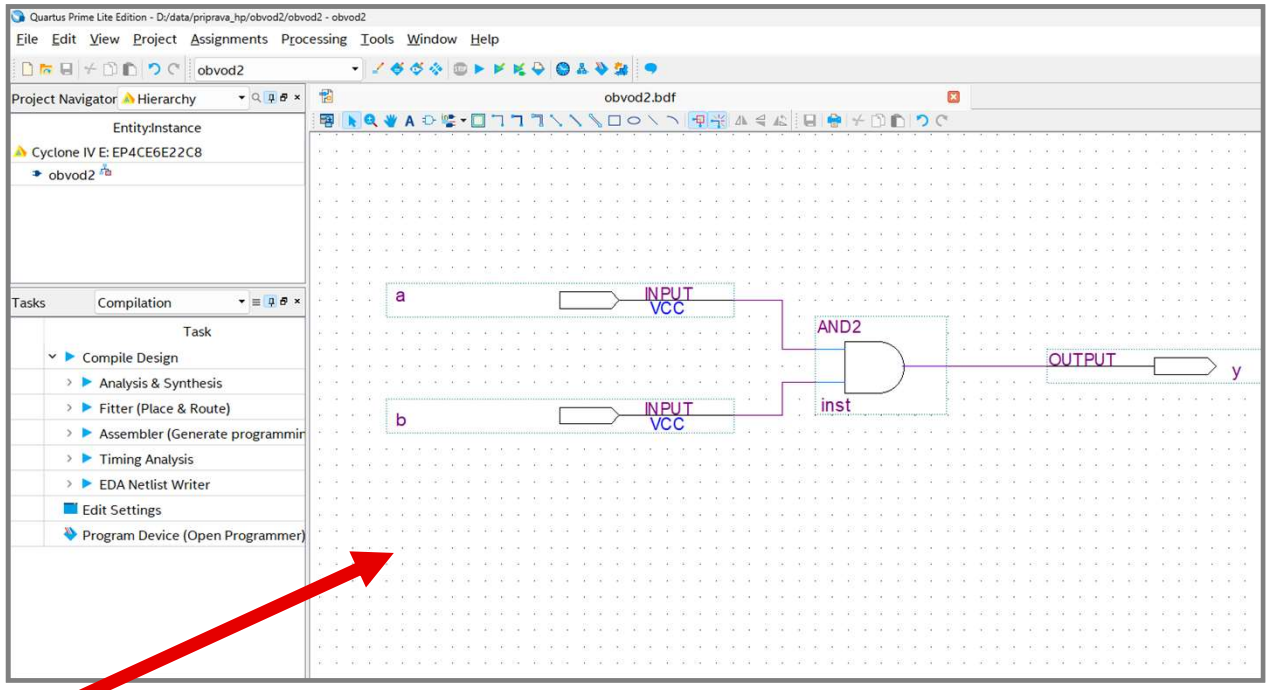


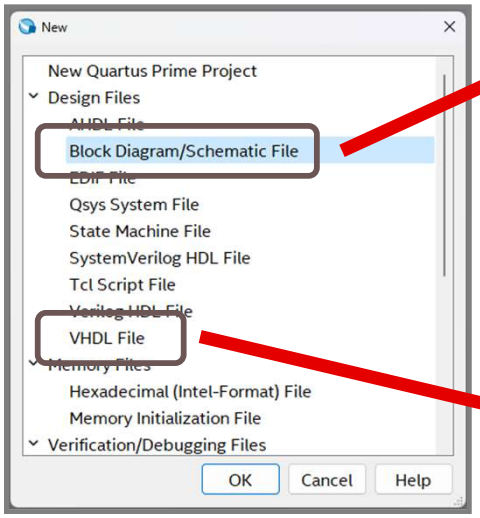


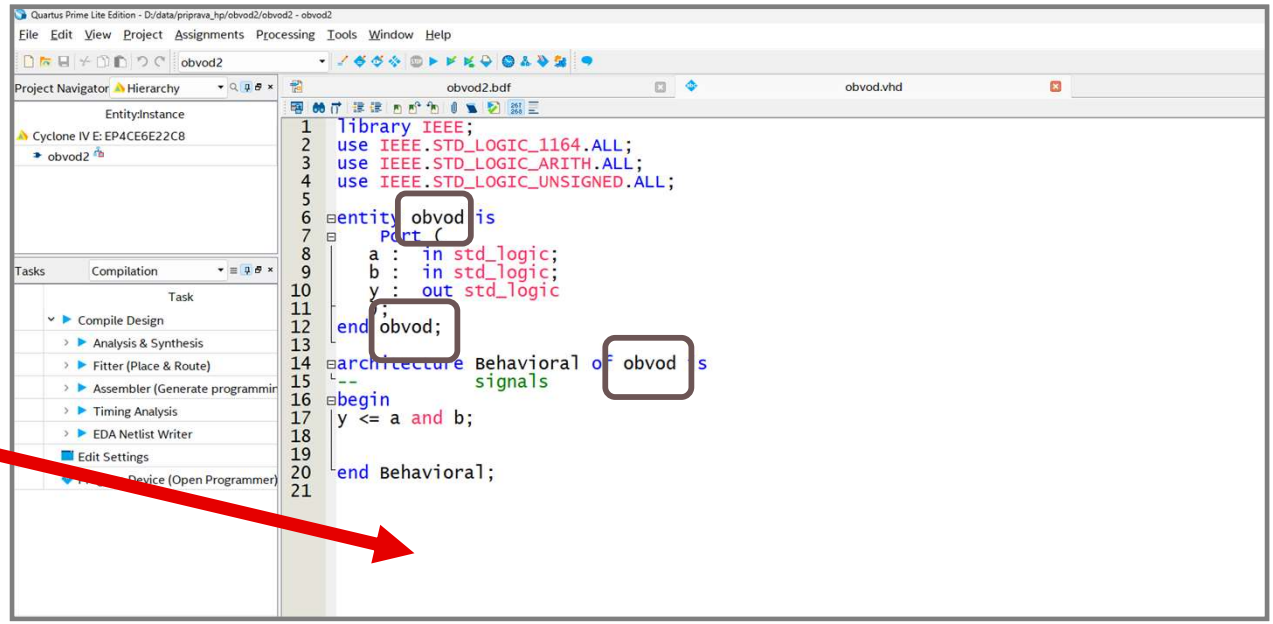
Cyclone IV E:
EP4CE6E22C8

2. Volba typu aplikace

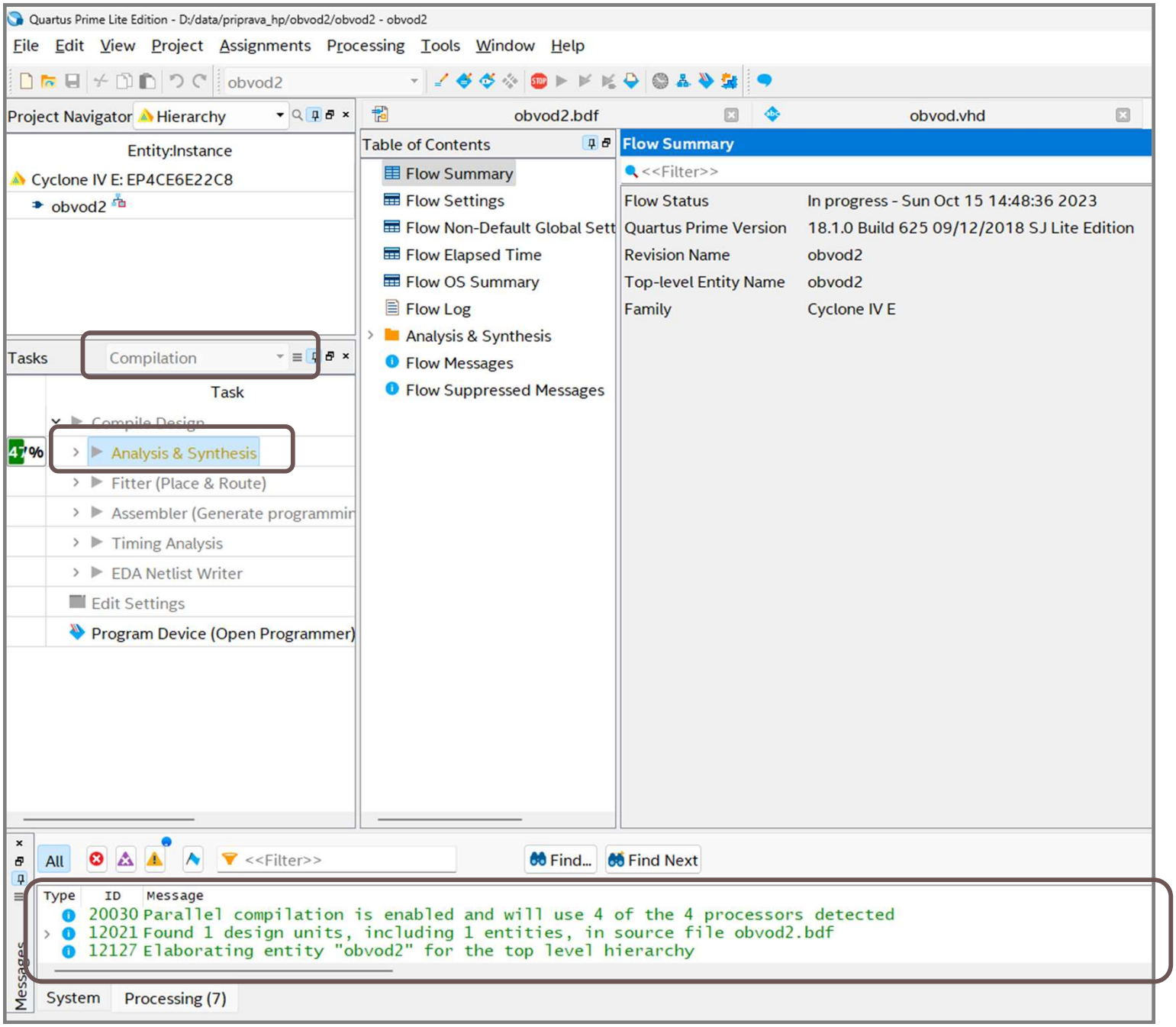




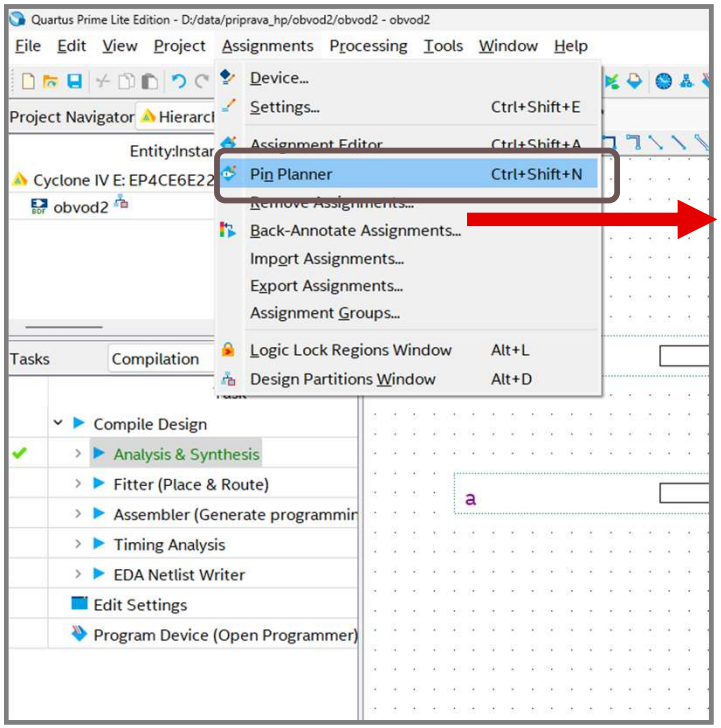


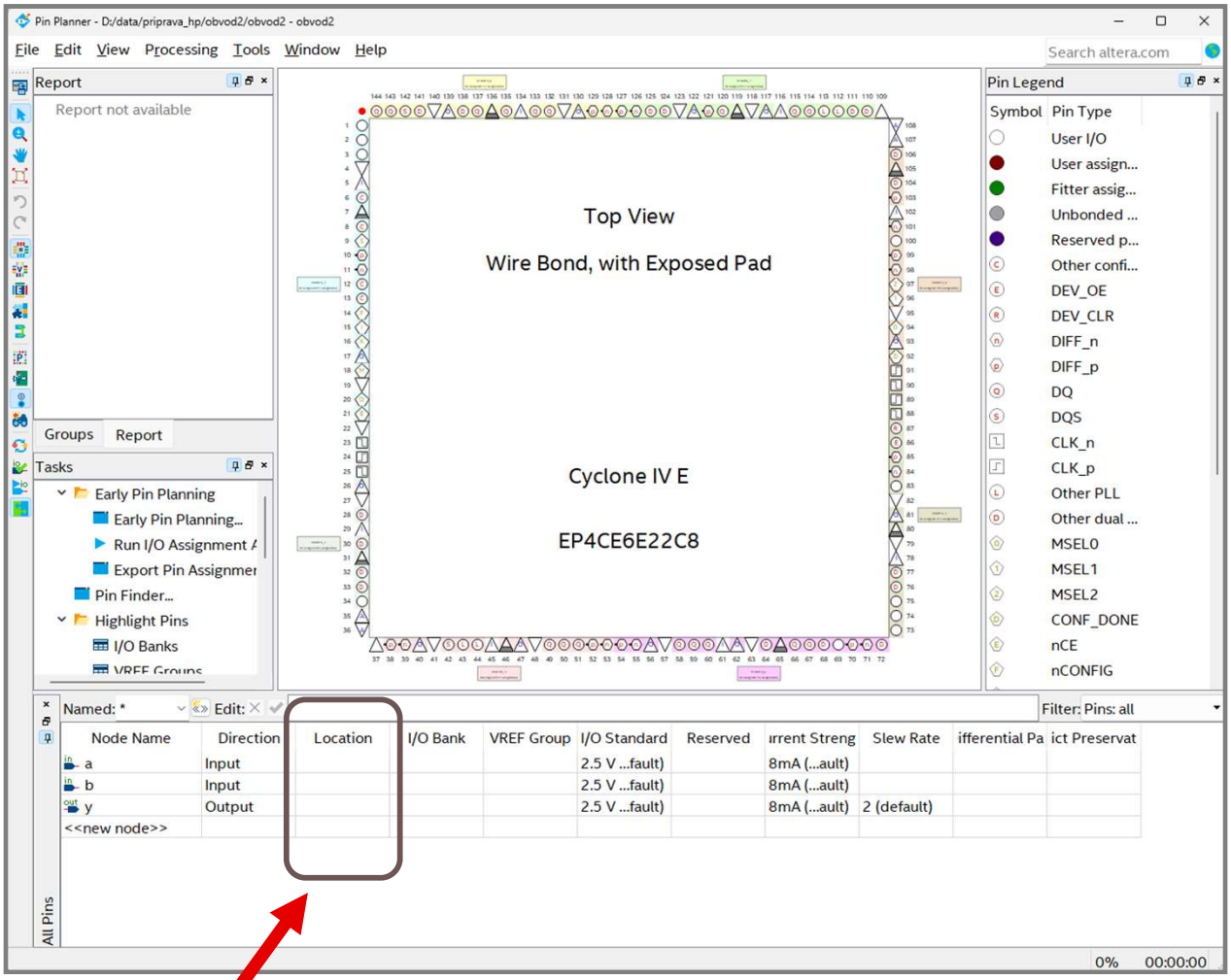


3. Kontrola syntaxe



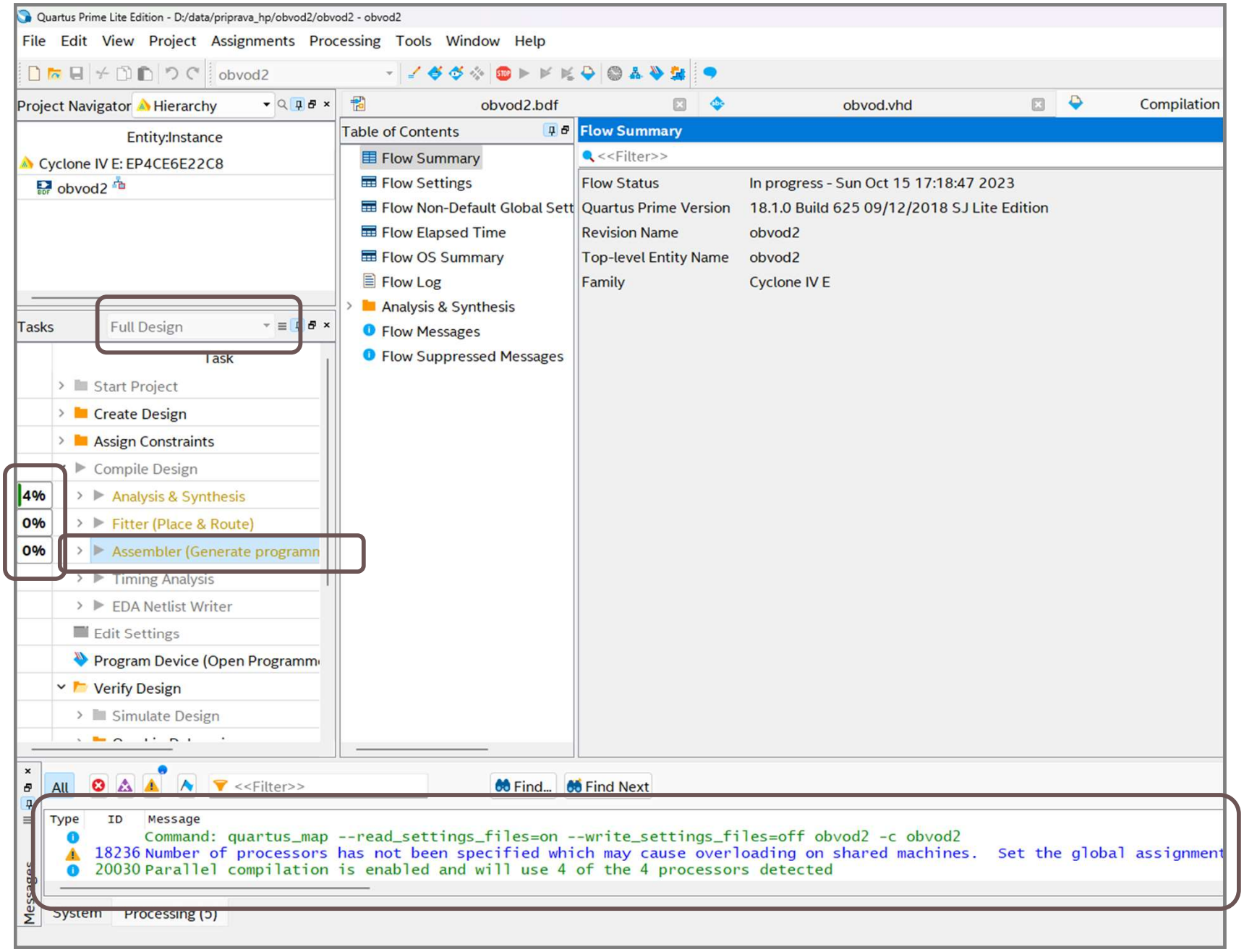
4. Připojení markerů k vývodům FPGA



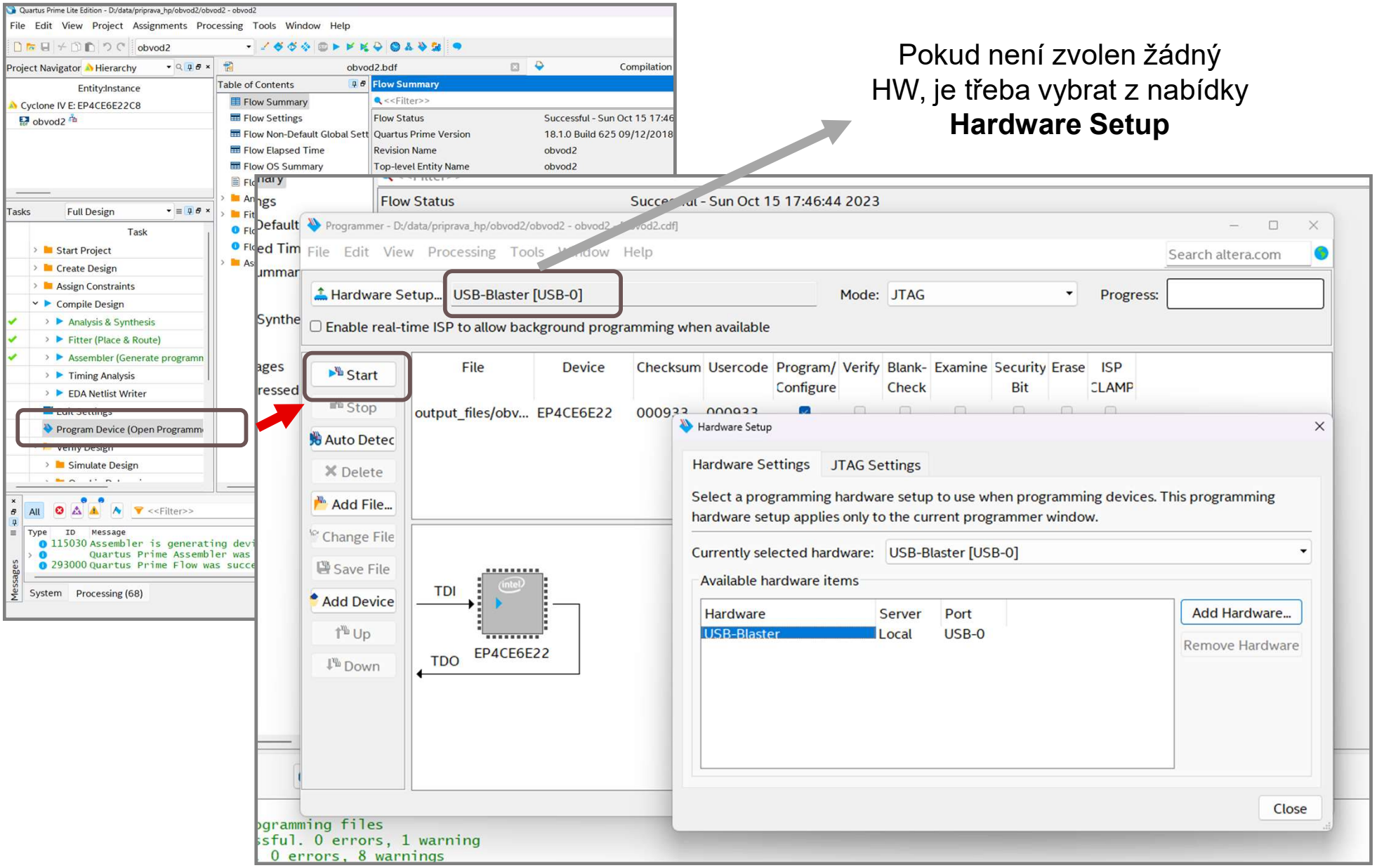


a – 88
b – 89
y - 87

5. Generování programovacího souboru



6. Nahrání do přípravku



Aplikace VHDL

Základní pojmy

Zdrojový soubor

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

entity logika is
    Port ();
end logika;

architecture Behavioral of logika is
        Signály
begin
        Definice
        struktury HW
end Behavioral;
```

Knihovny

Vstupy a výstupy

Základní pravidla

Jednoduchý vstup a výstup

```
a :  in std_logic;
```

Název: **a**
Směr: **in**

```
y :  out std_logic;
```

Název: **y**
Směr: **out**

Sběrnice vstup a výstup

```
adresa :  in std_logic_vector(7 downto 0);
```

Název: **adresa**
Směr: **in**
Šířka: **8 bitů**

```
data :  out std_logic_vector(3 downto 0);
```

Název: **adresa**
Směr: **in**
Šířka: **8 bitů**

Signály

```
citac :  std_logic;
citac :  std_logic:= '0';
citac :  std_logic_vector(4 downto 0)="10010";
citac :  std_logic_vector(x downto 0)=(others =>'0');
```

Název: **citac**
Šířka: **5**

Název: **citac**
Šířka: **libovolná**

Základní pravidla

Přiřazení <=

```
y <=  a and b;
```

Základní hradla: **and, or, xor, not**

Sloučení

```
b <= "0" & a & c(7 downto 1);
```

```
C šířka 6 bitů (bity 1-7)
A šířka 1 bit
B šířka 8 bitů
konstanta 2 bity
```

Rozdělení

```
b <= c(7);
```

```
C šířka 1 bit (7.bit)
B šířka 1 bit
```

Číselné soustavy

```
"0";
X"9";
```

Asynchronní podmínky

Select /When Statement

Select /When

```
with vstup select
        výstup <= hodnota1 when hodnota_vstupu1,
                  hodnota2 when hodnota_vstupu2,
                  hodnota3 when others;
```

Dekodér

```
x: vstup
y: výstup
```

```
with vstup select
        y <=      "0001" when "00",
                  "0010" when "01",
                  "0100" when "10",
                  "1000";
```

When /Else Statement

When /Else

```
výstup <= hodnota1 when vstup=hodnota_vstupu1 else
          hodnota2 when vstup=hodnota_vstupu2 else
          hodnota3;
```

Dekodér

```
x: vstup
y: výstup
```

```
y <=      "0001" when x="00" else
          "0010" when x="01" else
          "0100" when x="10" else
          "1000";
```

Synchronní podmínky

Case

Case

```
process(clk)
begin
        if ( clk'event and clk ='1') then
                case (<2-bit select>) is
                        when "00" =>
                                <statement>;
                        when "01" =>
                                <statement>;
                        when "10" =>
                                <statement>;
                        when "11" =>
                                <statement>;
                        when others =>
                                <statement>;
                end case;
        end if;
end process;
```

Case - dekodér

Case

x: vstup
y: výstup

```
process(x)
begin
        if ( clk'event and clk ='1') then
                case (x) is
                        when "00" =>
                                y <= "1000";
                        when "01" =>
                                y <= "1000";
                        when "10" =>
                                y <= "1000";
                        when others =>
                                y <= "1000";
                end case;
        end if;
end process;
```

If/Elsif/Else

If/Elsif/Else

```
process(clk)
begin
        if ( clk'event and clk ='1') then
                if <condition> then
                        <statement>
                elsif <condition> then
                        <statement>
                else
                        <statement>
                end if;
        end if;
end process;
```

If/Elsif/Else - dekodér

If/Elsif/Else

x: vstup
y: výstup

```
process(clk)
begin
        if ( clk'event and clk ='1') then
                if x="00" then
                        y <= "1000"
                elsif x="01" then
                        y <= "0100"
                elsif x="10" then
                        y <= "0010"
                else
                        y <= "0001"
                end if;
        end if;
end process;
```

Kombinační obvody

Process

```
process(<citlivostní seznam>)
begin
        --tělo procesu
end process;
```

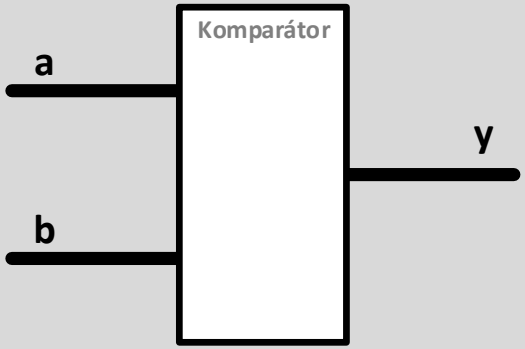
Proces – synchronní struktura

Citlivostní seznam – Proces se spustí, když dojde ke změně úrovně veličiny v citlivostním seznamu

Tělo procesu – V těle jsou příkazy popisující zapojení obvodu

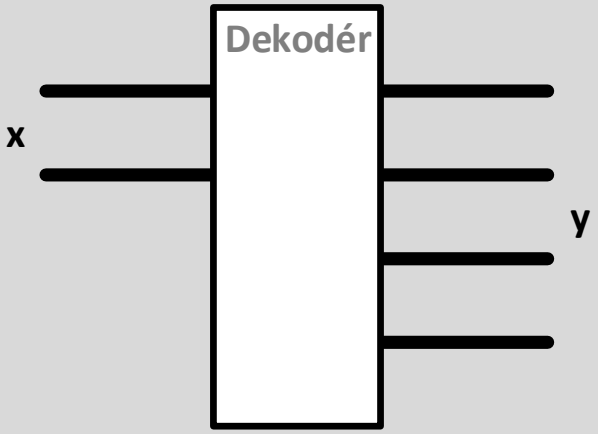
Komparátor - If/Elseif/Else

```
process(a,b,clk)
begin
        if ( clk'event and clk ='1') then
                if (a=b) then
                        b <= '1’;
                else
                        b <= '0’;
                end if;
        end if;
end process;
```



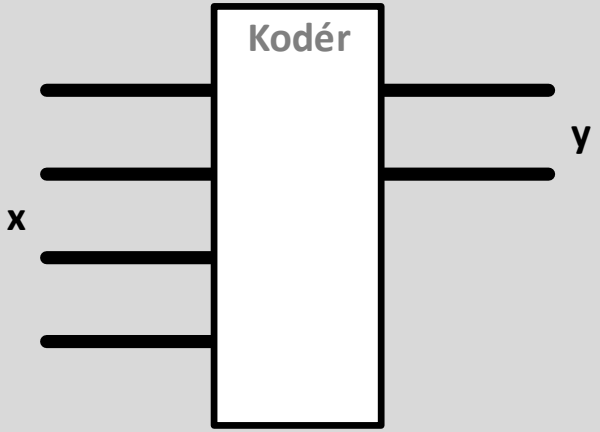
Dekodér - Case

```
process(x,clk)
begin
        if (clk'event and clk ='1') then
                case (x) is
                        when "00" =>
                                y <= "0001";
                        when "01" =>
                                y <= "0010";
                        when "10" =>
                                y <= "0100";
                        when others =>
                                y <= "1000";
                end case;
        end case;
end process;
```



Kodér - Case

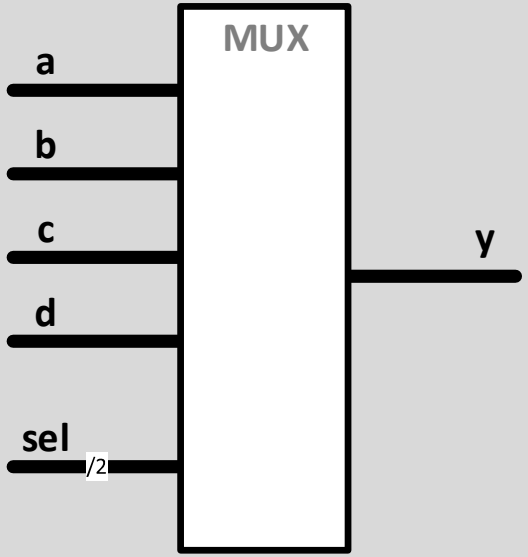
```
process(x,clk)
begin
        if (clk'event and clk ='1') then
                case (x) is
                        when "0001" =>
                                y <= "00";
                        when "0010" =>
                                y <= "01";
                        when "0100" =>
                                y <= "10";
                        when "1000" =>
                                y <= "11";
                        when others =>
                                y <= "00";
                end case;
        end case;
end process;
```



Multiplexer – různé konstrukce

When/Else

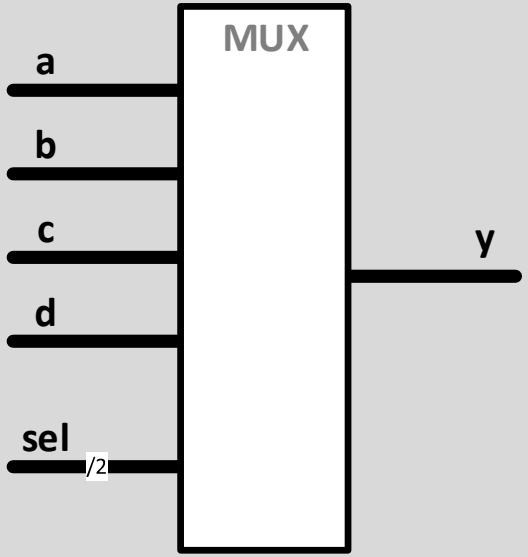
```
y <=    a when sel="00" else
        b when sel="01" else
        b when sel="01" else
        d;
```



Multiplexer – různé konstrukce

Select /When

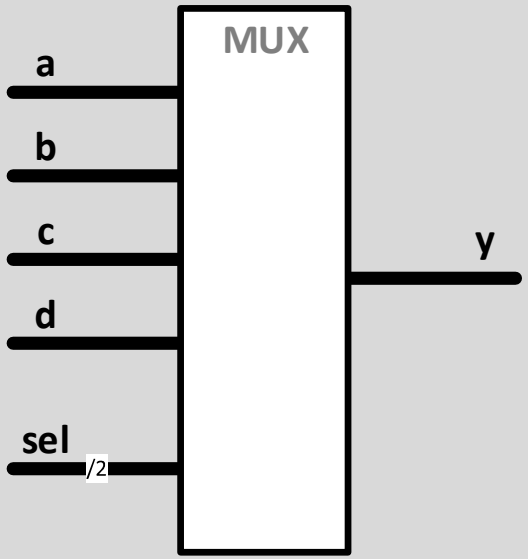
```
with sel select
        y <=    a when "00" else
                b when "01" else
                b when "01" else
                d;
```



Multiplexer – různé konstrukce

Synchronní/Case

```
process(a,b,c,d,sel)
begin
        case sel is
                when "00" =>
                        y <=a;
                when "01" =>
                        y <=b;
                when "01" =>
                        y <=c;
                when others
                        y <=d;
        end case
end process;
```



Multiplexer – různé konstrukce

Synchronní/If

```
process(a,b,c,d,sel)
begin
        if sel="00" then
                y <=a;
        elsif sel="01" then
                y <=b;
        elsif sel="10" then>
                y <=c;
        else
                y <=d;
        end case
end process;
```