Laboratorní cvičení z předmětu CITE

Podmíněné přiřazení, multiplexor

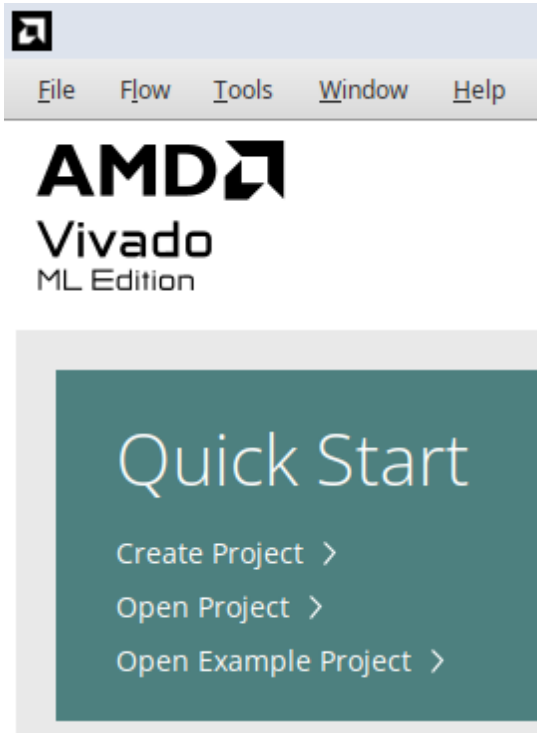
Martin Rozkovec, martin.rozkovec@tul.cz
Technická univerzita v Liberci | Fakulta mechatroniky, informatiky a mezioborových studií | ITE
Studentská 1402/2, 461 17 Liberec 1 | www.fm.tul.cz

1. Z kurzu předmětu na elearning.tul.cz stáhněte a rozbalte projekt LAB03.zip.

   Cesta k projektu nesmí obsahovat diakritiku, mezery nebo speciální znaky. Vhodné umístění je v laboratoři A107 vaše domovská složka, ve kterém si můžete vytvořit podadresář s vaším jménem bez diakritiky.

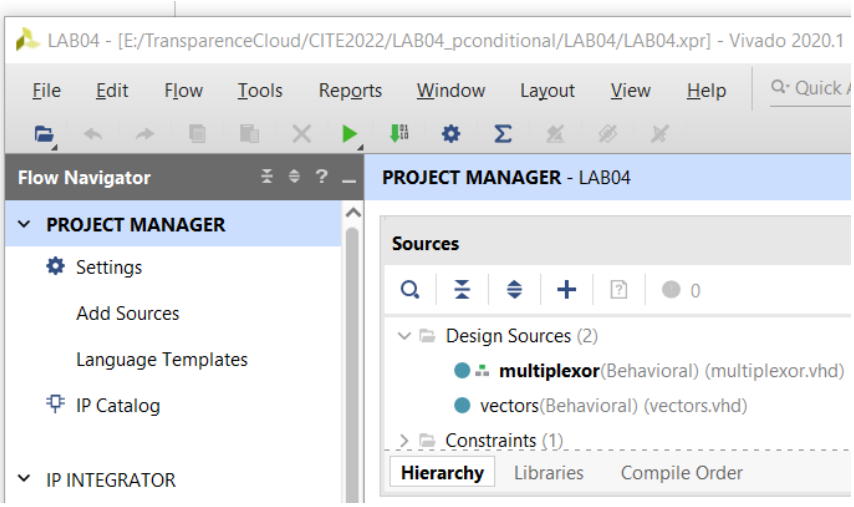
2. Otevřete Vivado – dvakrát klikněte na spouštěč na ploše:

3. V úvodním okně aplikace klikněte na Quick Start – Open Project:



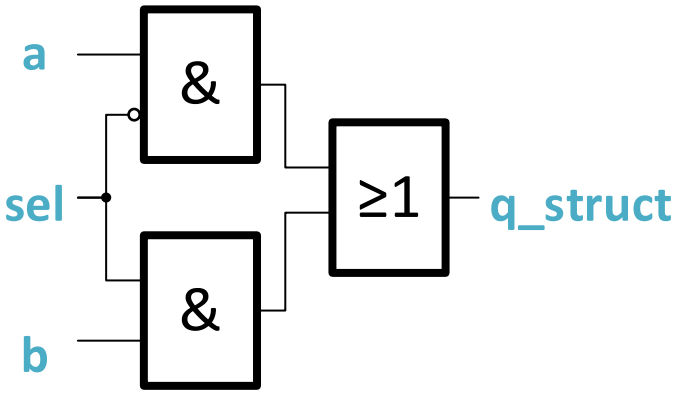
4. Otevřete soubor LAB04.xpr uvnitř dekomprimované složky.

5. Otevře se prostředí Vivado. Otevřete kód souboru multiplexor.vhd:

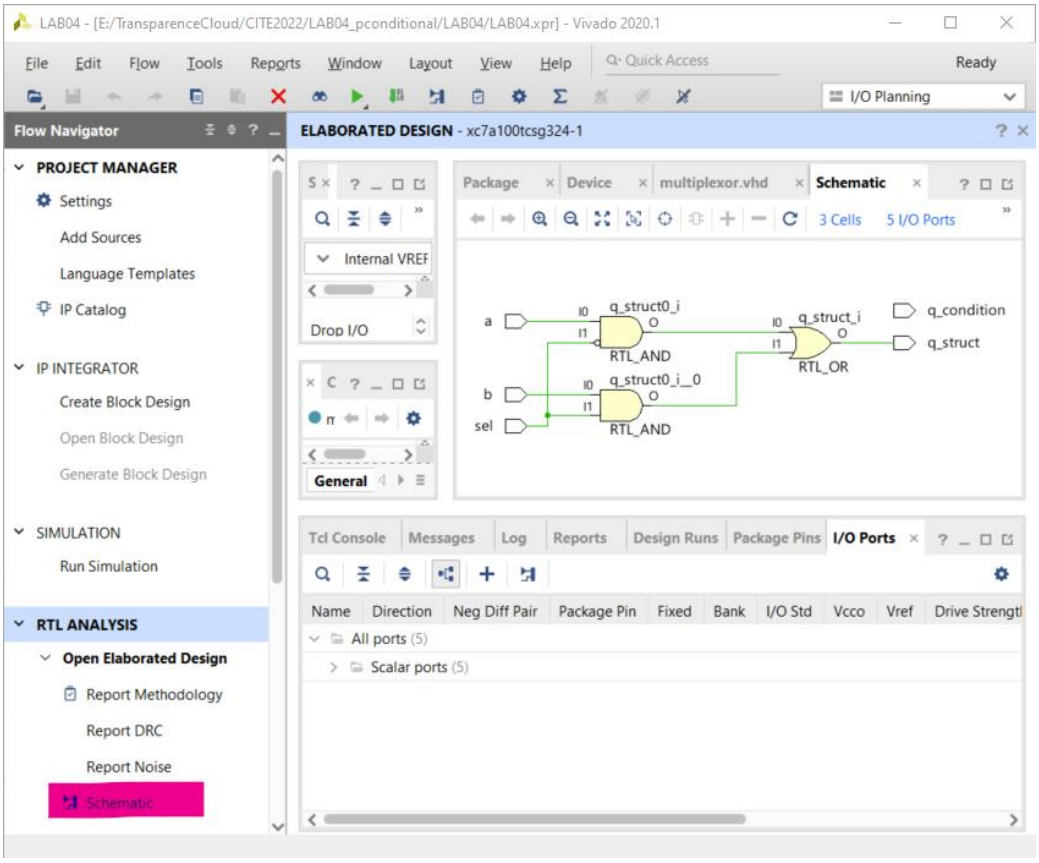


Multiplexor

1. Pomocí výrazu nepodmíněného přiřazení, který jste procvičili v minulém cvičení vytvořte multiplexor s dvojicí vstupů **a**, **b**, výběrovým signálem **sel** a výstupem **q_struct**.



2. Ověřte, zda schéma odpovídá předpokladu:



3. Multiplexor jsme popsali strukturně. Můžeme popsat i jeho chování:
    a. Chceme na výstup **q** přiřadit proměnnou **a**, když je vstup **sel** roven logické **0**, jinak přiřadíme **b**.
4. Seznamte se syntaxí příkazu podmíněného přiřazení:

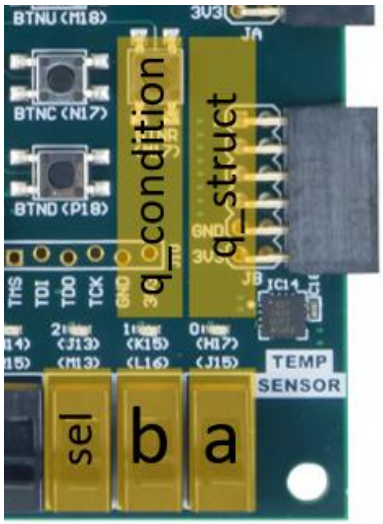
```
cíl <= výraz_0 when podmínka_0
[else výraz_1  when podmínka_1
 else výraz_2  when podmínka_2
 ...]
[else výraz_n];
```

- Příkaz podmíněného přiřazení je alternativou podmínky if v programovacích jazycích, které už znáte.
- **Nepovinné** části příkazu jsou uzavřeny v hranatých závorkách [ ]. První nepovinná část se neuvádí, pokud popisujeme binární rozhodování. Druhou nepovinnou část (tj. poslední větev **else** bez podmínky) uvádíme **vždy**, když popisujeme **kombinační logiku**.
- **Podmínka** je booleovská (typ **boolean** s hodnotami **true** a **false**). Podmínku můžeme definovat explicitně pomocí relačních operátorů (=, /=) nebo využít implicitní zápis (pouze

s proměnnou typu **std_logic**). Porovnávat lze pouze kompatibilní datové typy, proto píšeme na pravou stranu jednoduché uvozovky. Implicitní zápis je konvertován na podmínku sel = '1'.

```
sel = '1' nebo sel /= '0' nebo sel
```

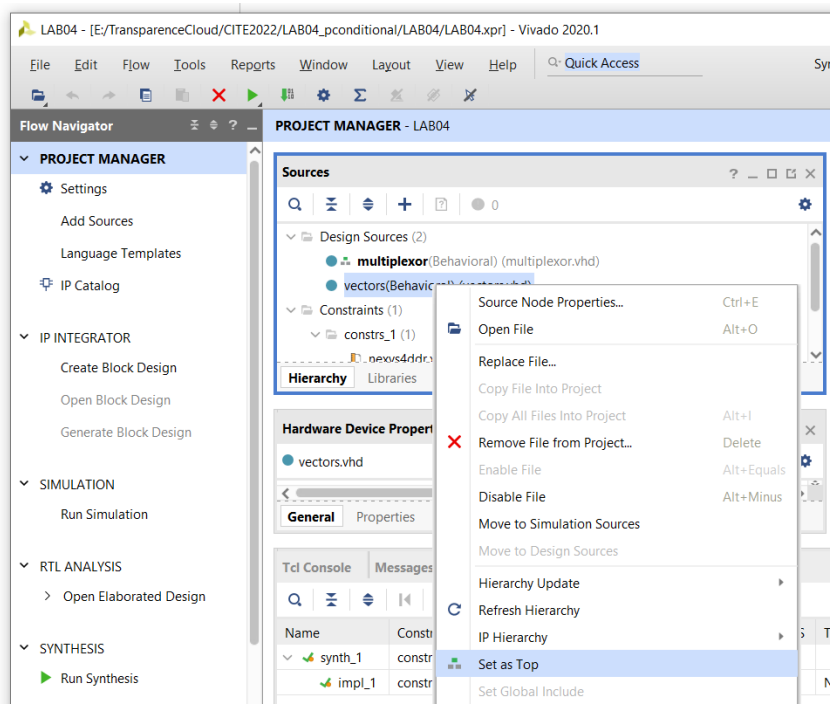
5. Použijte příkaz podmíněného přiřazení na vytvoření popisu funkce multiplexoru. Přiřazujte do výstupu **q_condition**.
6. Prozkoumejte schéma obvodu v okně Schematic RTL analýzy. Behaviorálně popsaný multiplexor má značku multiplexoru.
7. Ověřte, že se oba multiplexory chovají shodně:
    a. Pro nahrání do obvodu:
        i. Klikněte na Generate Bitstream v menu Flow.
        ii. Po ukončení procesu zvolte Open Hardware Manager.
        iii. Připojte desku a zapněte ji.
        iv. Klikněte v zelené liště na Open target – Auto Connect.
        v. Klikněte v zelené liště na tlačítko Program a potvrďte.



Behaviorální popis ve většině případů odstraňuje zdlouhavou tvorbu pravdivostní tabulky a minimalizaci

STD_LOGIC_VECTOR

1. Přepněte Vivado na zpracování souboru vectors.vhd:
    a. Pravým tlačítkem myši klikněte na položku vectors.vhd v Project Manager \ Sources a zvolte Set as Top.

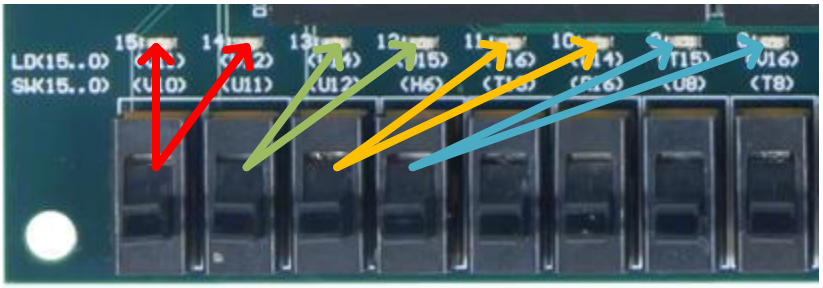


2. Povšimněte si:
    a. Datový typ vstupu i výstupu se změnil na std_logic_vector.
        - **std_logic_vector** je pole hodnot **std_logic**.
        - Velikost pole uvádíme pomocí rozsahu - **range** a to převážně ve formě (**High downto Low**) nebo výjimečně (**Low to High**). Počet prvků pole je **High** – **Low** + **1**, nehledě na směr.
    - Do vektoru zapisujeme konstanty v dvojitých uvozovkách, např: **V <= "0111"**. Bázový typ vektoru je znak (**'0'**); vektor je svého druhu **řetězec** znaků.
    - Chceme-li zapisovat v jiné, než implicitní binární číselné soustavě volíme [N]**D** pro desítkovou, [N]**X** pro šestnáctkovou a [N]**O** pro osmičkovou. Zápis doplňujeme nepovinným počtem bitů [N], které budou z řetězce použity. Hodnota konstanty v řetězci nesmí být po převodu do binární soustavy krácena. Např:

```
    signal x : std_logic_vector(4 downto 0) := 5D"30";
    signal y : std_logic_vector(3 downto 0) := X"F";
```

- Chceme-li všechny prvky nastavit na stejnou hodnotu, můžeme využít části příkazu **agregátu**. Jelikož jsou prvky pole typu std_logic, hodnotu dáváme do jednoduchých uvozovek, např: **LEDs <= (others => '0');** Více se o agregátech dozvíte ze skript.
- Prvky pole lze indexovat pomocí kulatých závorek do kterých zapíšeme:
    - **Integer**, chceme-li indexovat prvek pole, např: **LEDs**(4) <= **‚0‘**. Takový prvek je typu std_logic.
    - **Range**, chceme-li indexovat část pole, např.: **LEDs**(2 downto 1) <= **switches**(1 downto 0)**.** Rozměry výrazu na levé a pravé straně musí vždy odpovídat.
- Vektory lze řetězit pomocí operátoru slučování **&**. Délka vektoru na levé i pravé straně musí být rovna. Jakou hodnotu dosadíte za otazník **?**
    - LEDs <= **"0111" & '0' & switches(15 downto ?) & "111";**
    - (Jak se přijde na odpověd? Sčítáme délky slučovaných prvků na pravé straně a porovnáváme je s délkou cíle přiřazení…)

3. Proveďte následující přiřazení. Každé přiřazení pište na nový řádek a zakončete středníkem:
    a. Na **první dvě diody** přiřaďte **log. 0**. Na levé straně použijte výběr pomocí range, na pravé použijte příkaz **agregátu**.
    b. Na diody **2** a **3** přiřaďte **log. 1.**
    c. Na další **4** diody přiřaďte první čtyři přepínače. Využijte indexace pomocí **range** na obou stranách.
    d. Na posledních 8 diod přiřaďte 4 přepínače tak, že každý přepínač bude rozsvěcet dvě sousední LED. Na levé straně využijte indexaci pomocí **range**, na pravé využijte příkaz slučování vektorů **&**. Výsledek:



4. Vygenerujte bitstream a návrh vyzkoušejte na vývojové desce.

Multiplexor II

5. Přepněte Vivado na zpracování souboru **multiplexor.vhd** (pravé tlačítko na soubor **multiplexor.vhd** a zvolte **Set as Top**)
6. Upravte entitu tak, aby popisovala multiplexor, který přepíná 8 vstupů:
    a. Odstraňte všechny stávající porty.
    b. Přidejte **vstupní** datový port **d** typu **std_logic_vector** o délce **8**.
    c. Přidejte výběrový **vstupní** port **s** typu **std_logic_vector**. **Délku vstupu určete dle vstupu d.** Kolik bitů je potřeba k rozlišení všech možností vstupu **d**?
    d. Přidejte výstupní port **q** typu **std_logic**.
    e. **Pomocí příkazu podmíněného přiřazení popište multiplexor.**
    f. Zobrazte schéma multiplexoru.
    g. Získaný multiplexor otestujte na vývojové desce. Rozložení ovládání je na obrázku níže: