Návod je sice vytvořen pro starou verzi 7.2, ale plně poslouží k vytváření projektů ve verzi 17.1.

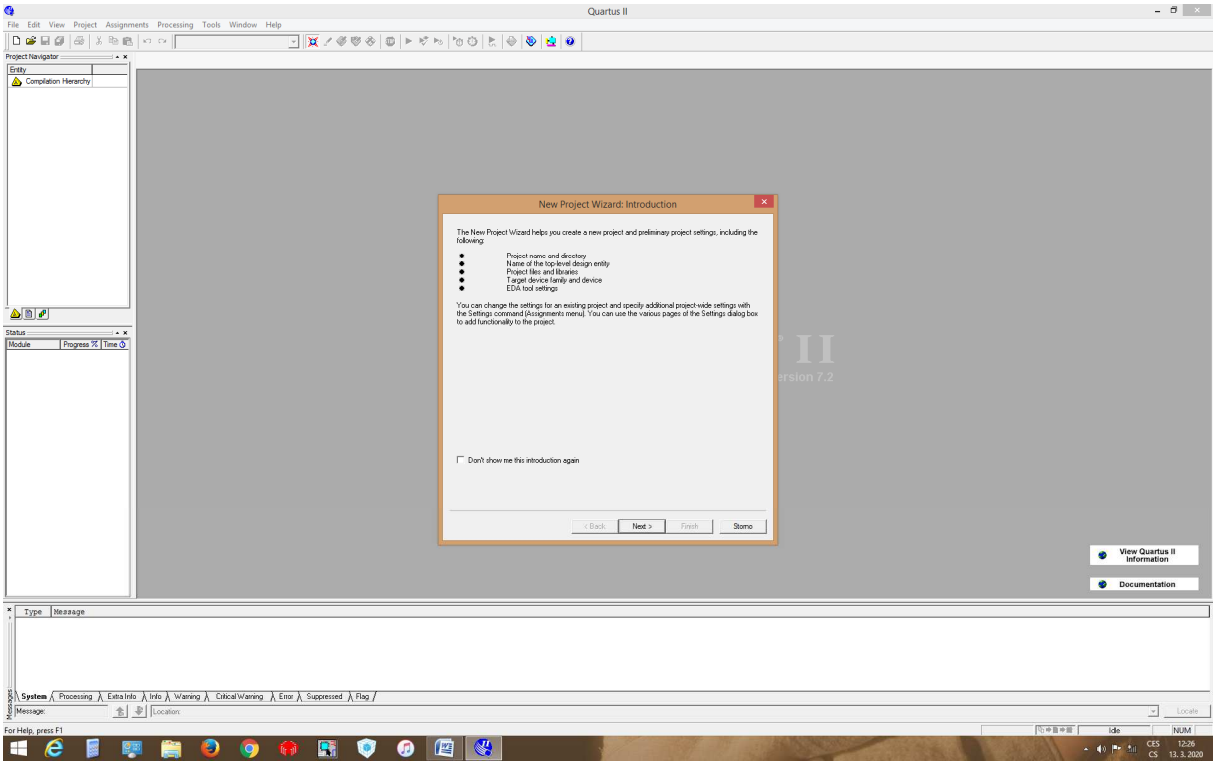
Začátek práce v systému Quartus II. Na obrazovce by jste měli mít následující ikonu

Quartus II 7.2sp3
Web Edition.lnk

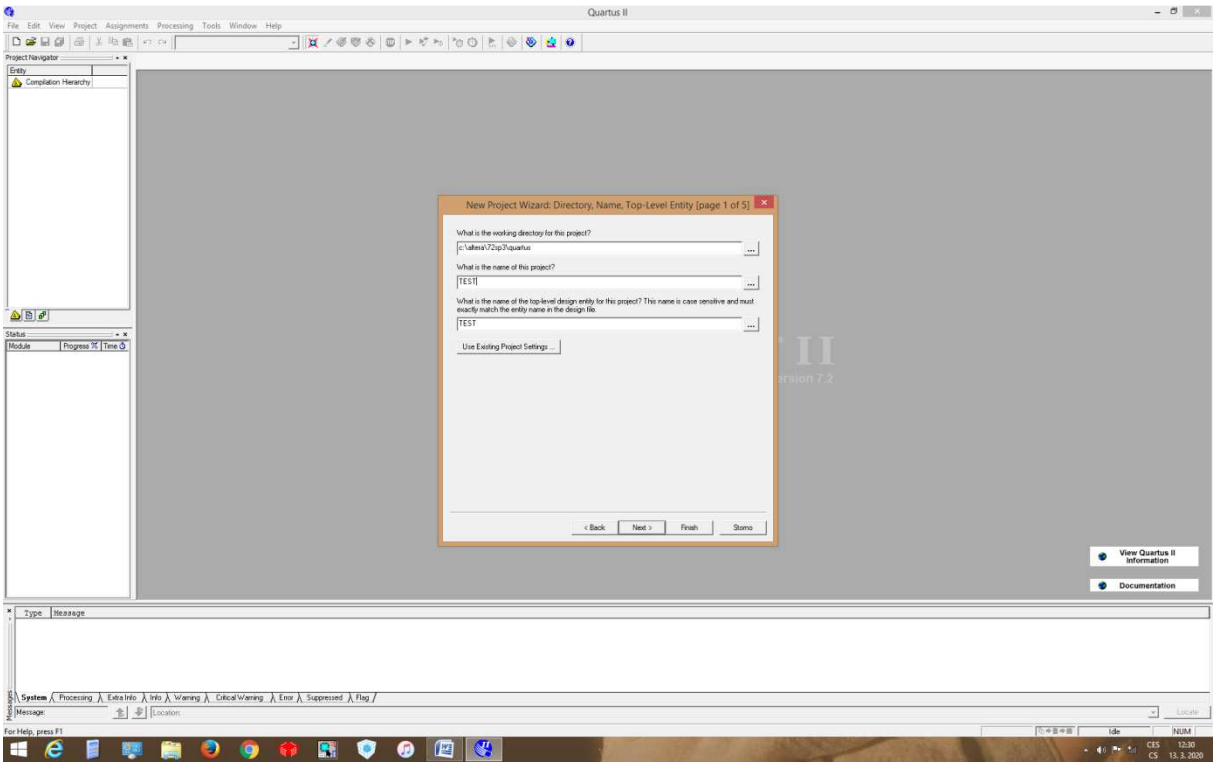
Poklepáním na ikonu se program rozběhne a bude Vás informovat , že se nemůže spojit s webem Altera, dáte OK. Po nastavení práce s 30 denní verzí budete mít na obrazovce následující stav.

Návrh a testování obvodů zadaných schématem

1) V položce **File** - čtvrtá položka **New Project Wizard** v následujícím okně dáte **NEXT**

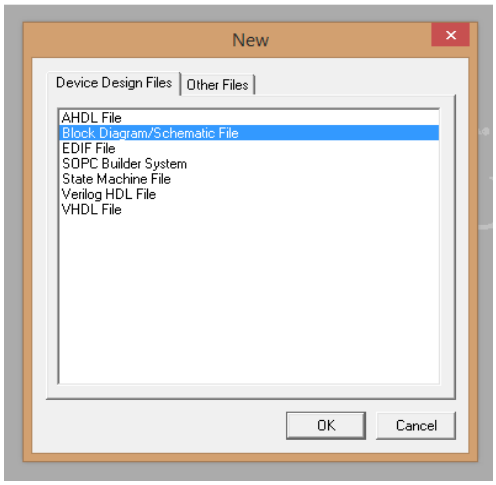


2) V dalším okně zadejte název projektu a jeho umístění a dejte **NEXT**

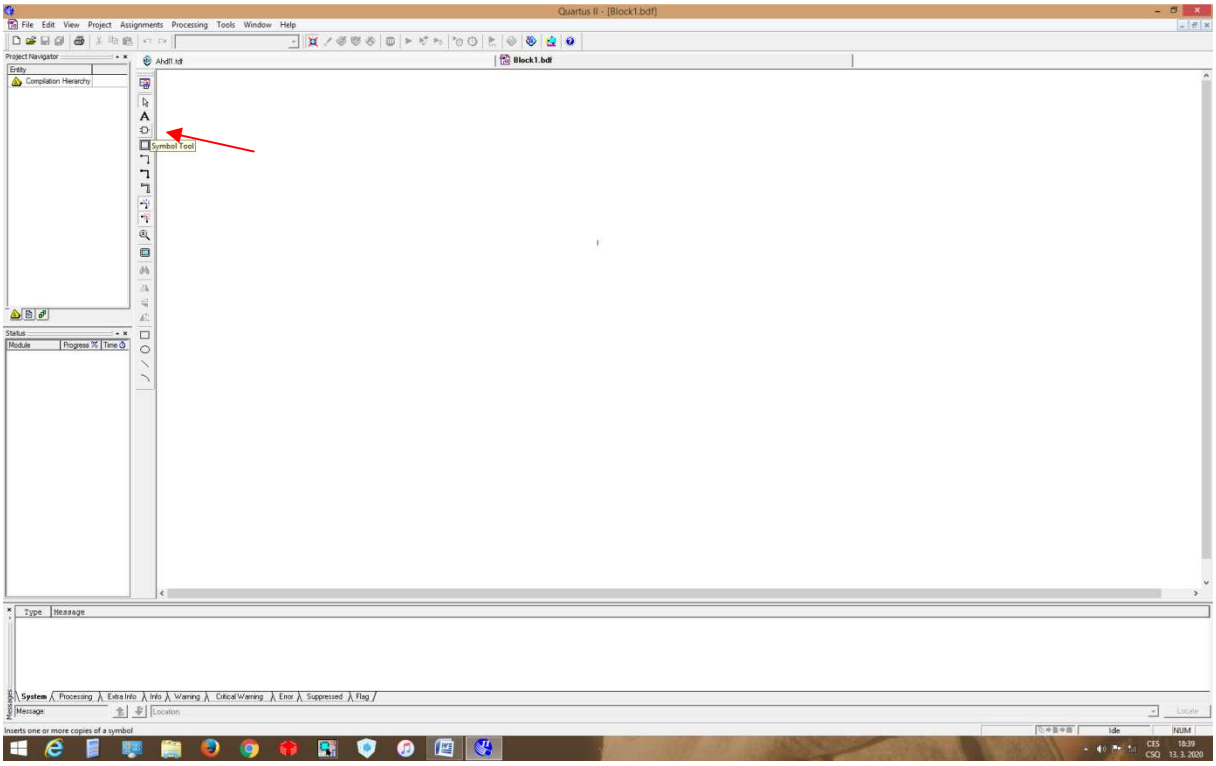


3) V dalším okně zadejte **NEXT** a v dalším zvolte Family **MAX3000A** obvod např. **EPM3032ALC44-4**
4) V dalším okně **NEXT** a následujícím **Finish**.

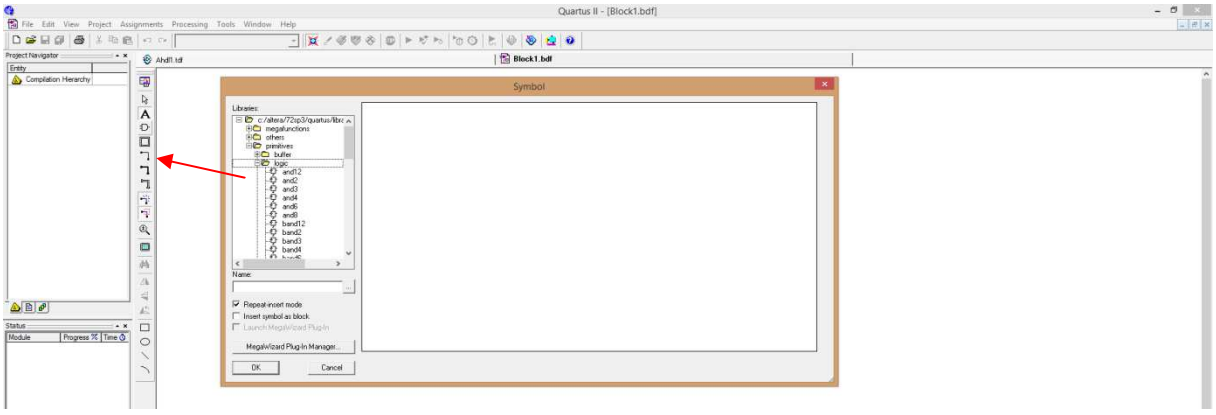
5) V položce **File** - první položka **New** vyberte druhou položku viz. obrázek a potom **OK**



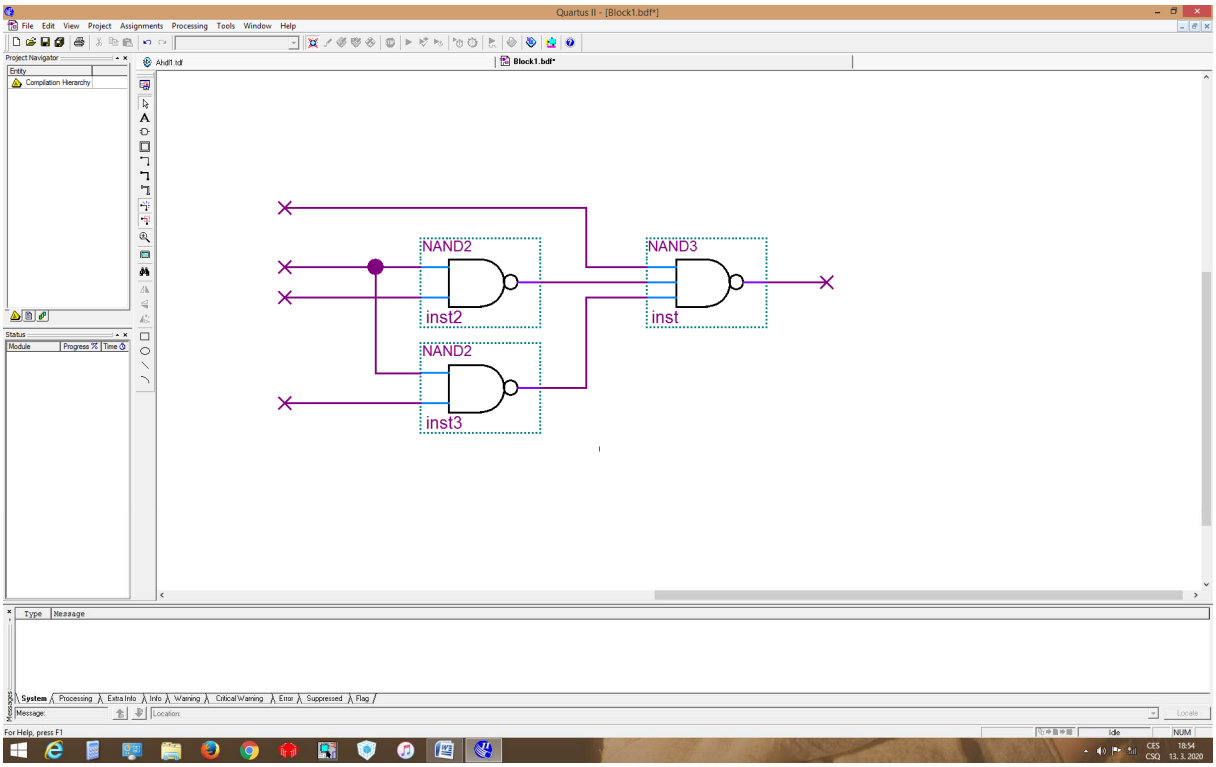
6) Získáte následující obrazovku



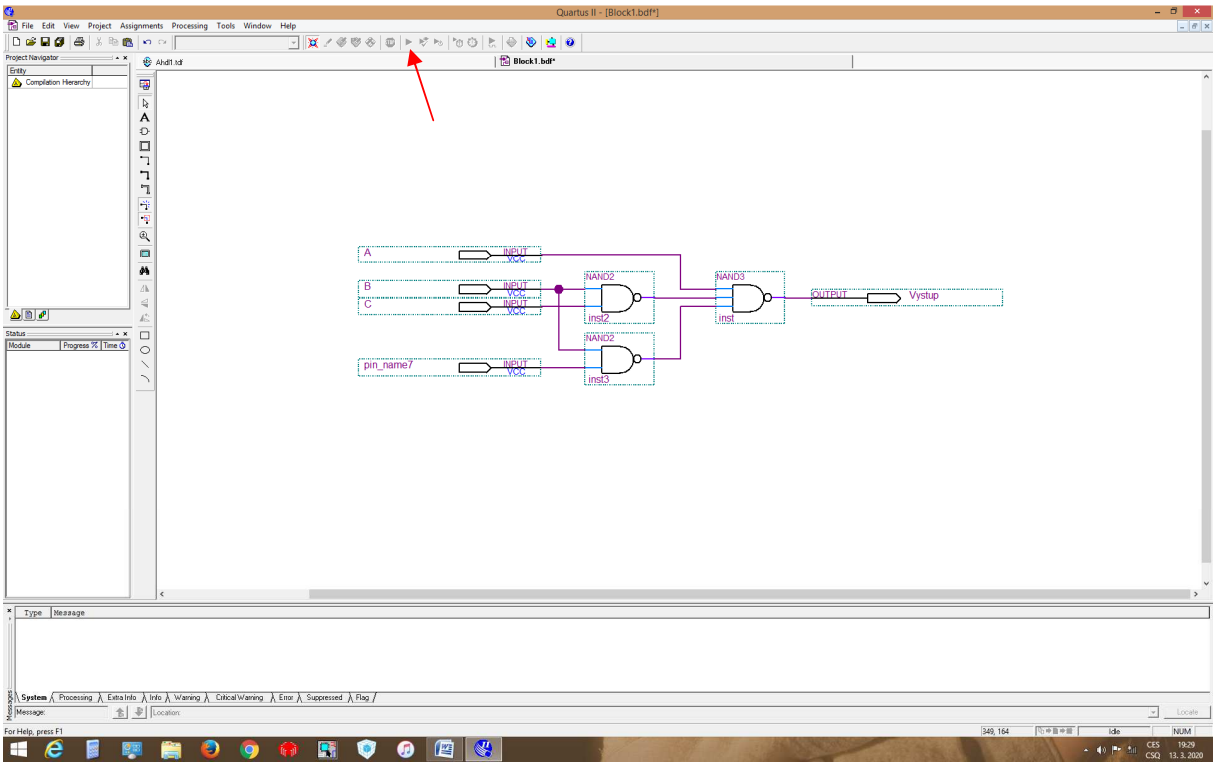
7) Do velké bíle plochy začnete kreslit schéma. Základní logická hradla jsou pod ikonou hradla. Rozevřené menu je zobrazeno na následujícím obrázku.



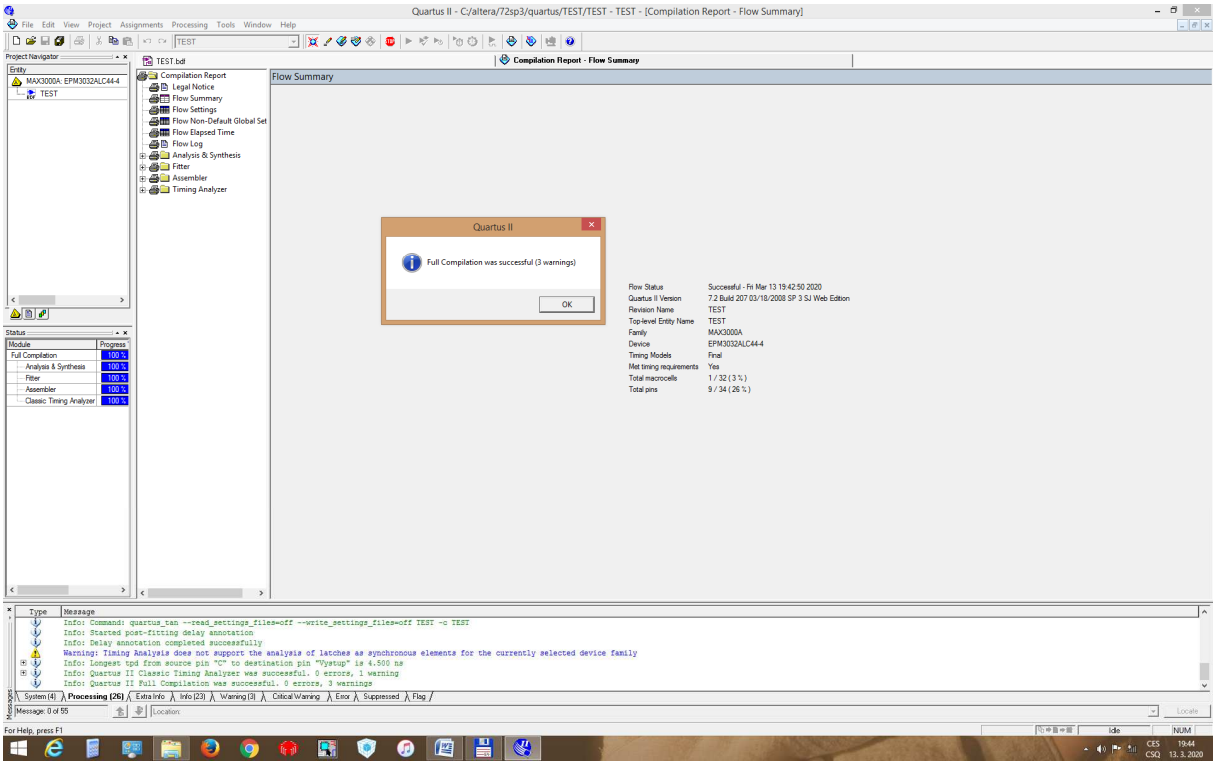
8) Vybereme potřebná hradla a umístíme jej na plochu. Přistoupíme k propojení jednotlivých obvodů pomocí ikony zobrazené na předchozím obrázku (propojovací čára) a dostaneme se do následujícího stavu.



9) Nyní musíme definovat vstupní a výstupní proměnné. Ty najdeme pod ikonou hradla v položce **primitives - pin.** Systémem nadefinované jméno proměnné můžeme změnit jak je ukázáno na obrázku.

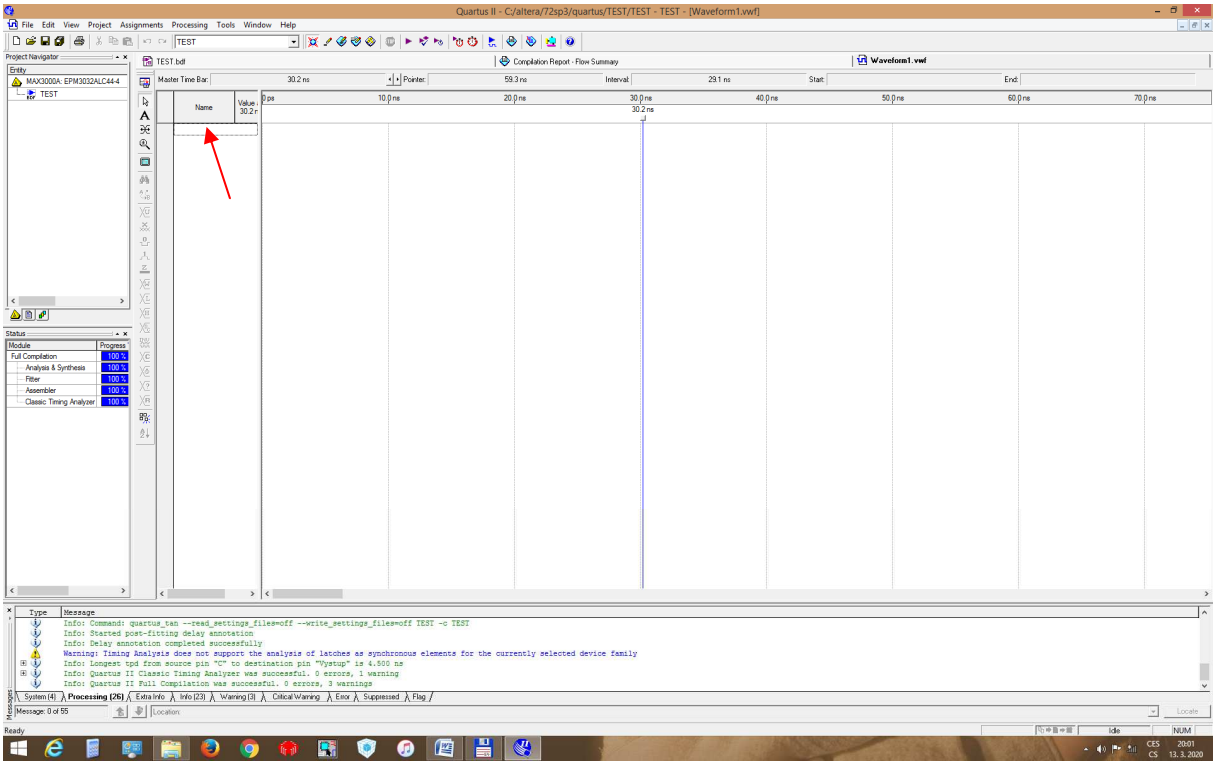


10) Vytvořené schéma uložíme. Následně provedeme překlad stiskem ikony ukázané na předchozím obrázku. Je-li vše v pořádku, pak získáme následující obrazovku.

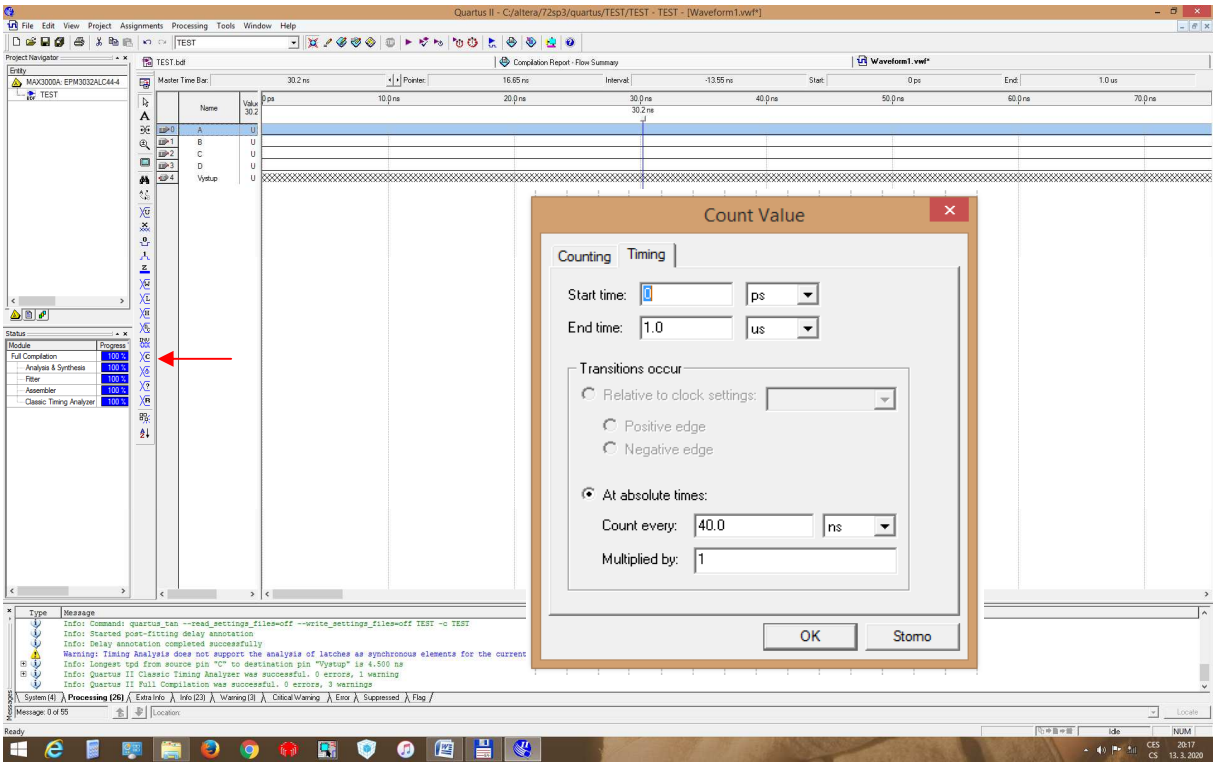


Vyjma informace o úspěšném překladu (varování si nemusíme všímat) získáváme první informace o velikosti návrhu. V uvedeném příkladu vidíme, že obvod k realizaci potřebuje 1 makrocell (3%) a 9 vývodů (5 pro realizaci obvodu a 4 pro programování obvodu) tj. 26%.

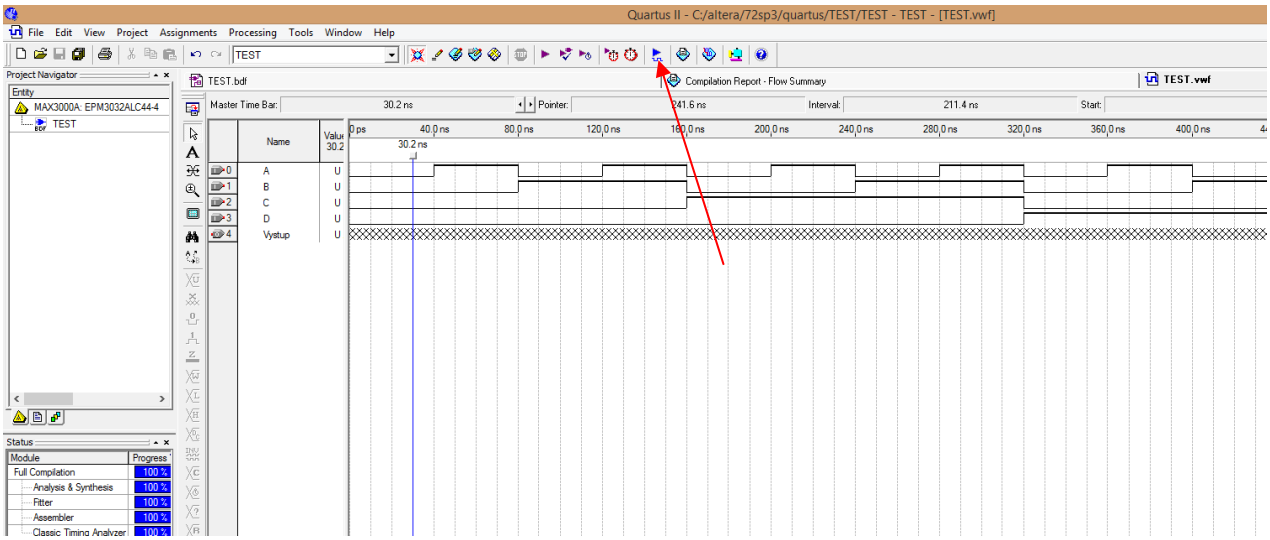
K ověření chování obvodu musíme vytvořit testovací soubor. Přímo v projektu začneme v položce **File - New - Other files - Vector Waveform File.** Získáme následující obrazovku.



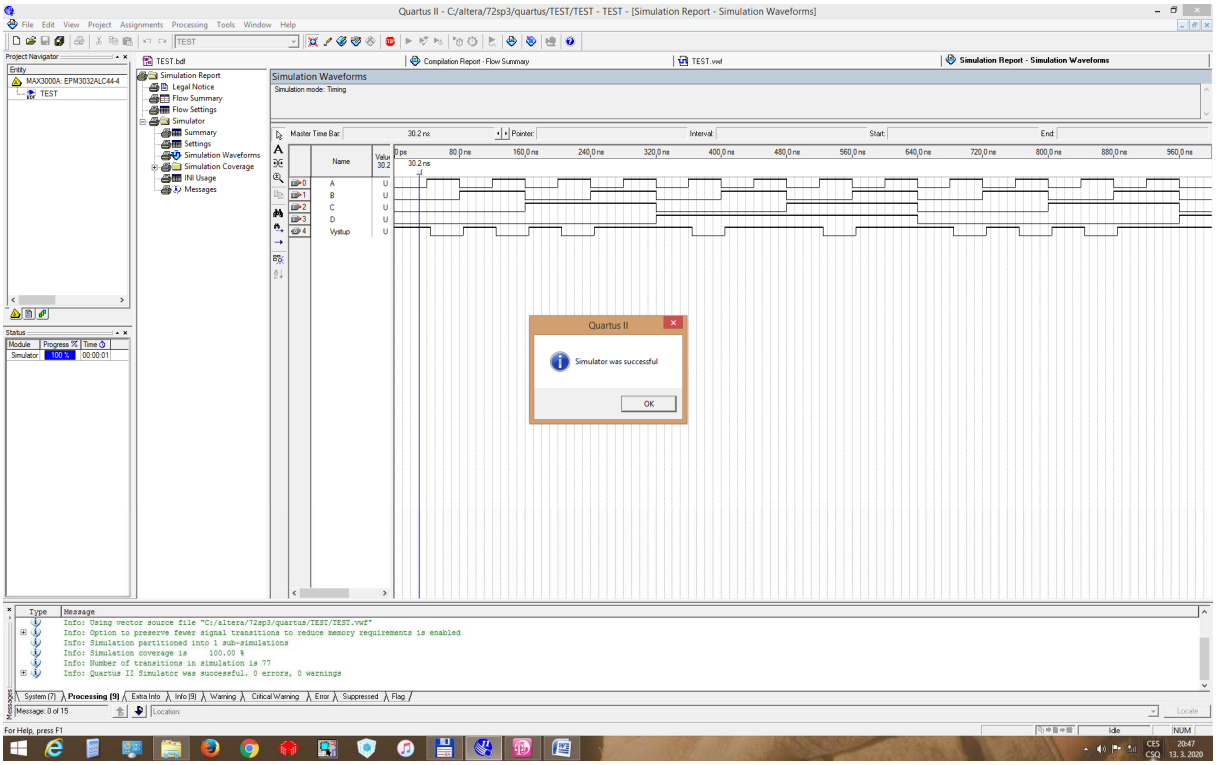
Jednotlivé proměnné pro testování můžeme vkládat postupně přes okénko zobrazené na předchozím obrázku. V okénku **Insert Node or Bus** vložíme jméno proměnné a dáme OK. stejným způsobem i ostatní proměnné. Nebo ve stejném okénku půjdeme na položku **Node finder - List**. Vybereme potřebné proměnné a přes ikonu > je přesuneme do pravého okna. Okno uzavřeme **OK**. Po tomto kroku bude obrazovka vypadat následovně.



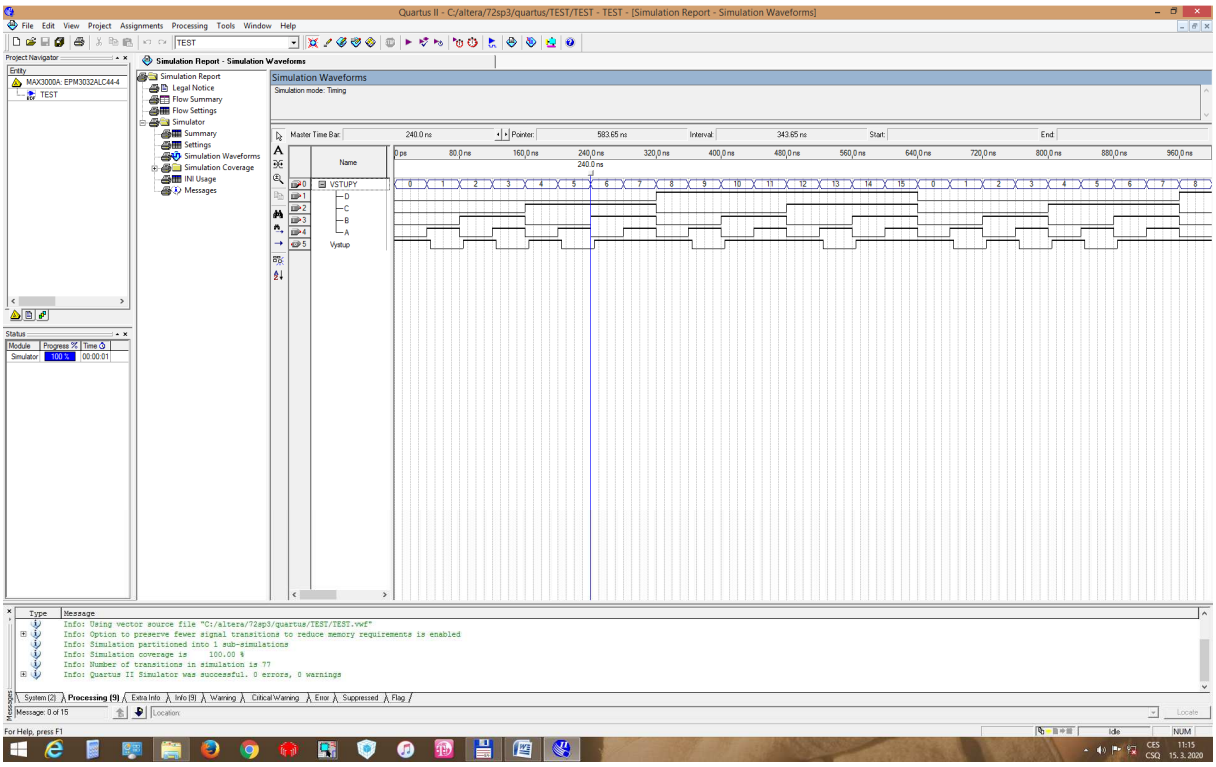
Nyní Pomocí ikonek můžeme graficky nastavit vstupní proměnné. Osobně za nejjednodušší považuji využit ikonu pro nastavení hodinového signálu (viz. předchozí obrázek). Zvýrazníme proměnnou (nejlépe s nejmenší váhou) a zmáčkneme ikonu na kterou ukazuje šipka. V okně **Count Value** nastavíme periodu změn (s ohledem na rychlost obvodu doporučuji začít) na 40ns. Pro proměnnou s vyšší váhou volte 80ns atd. Ve stavu z následujícího obrázku uložíme soubor. Ověření proběhne stiskem zobrazené ikony.



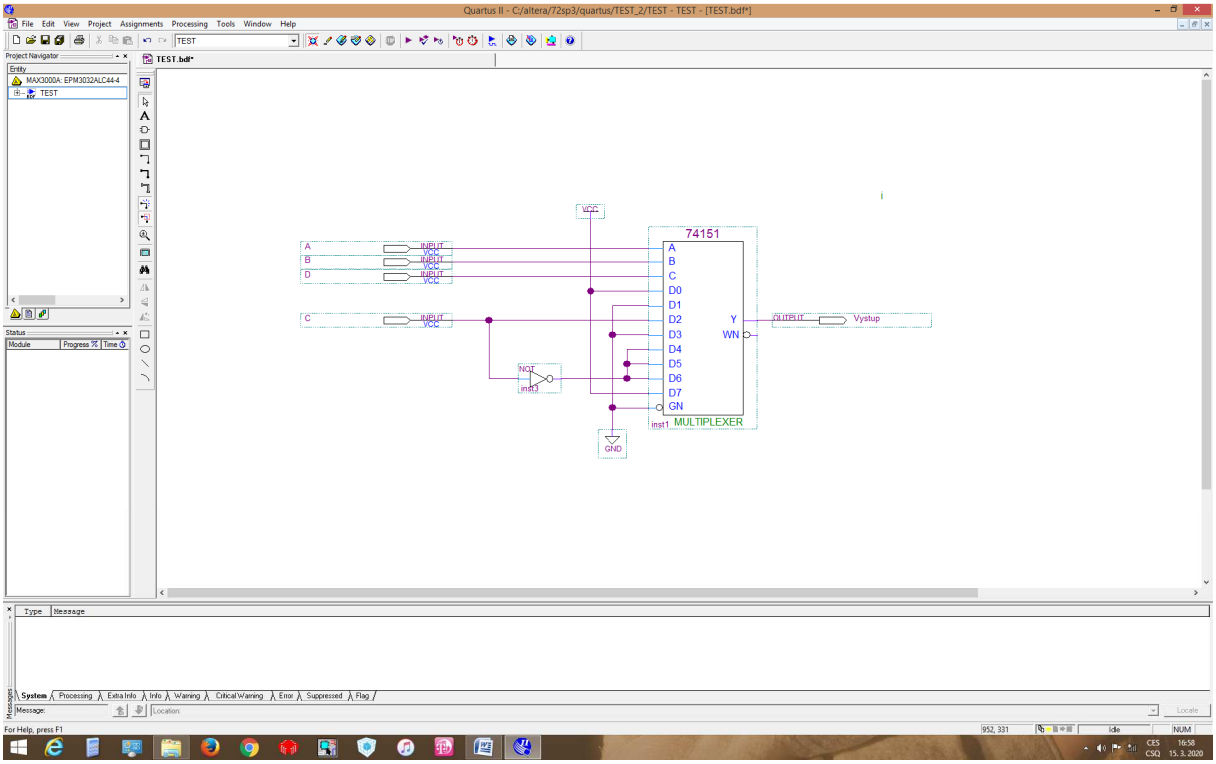
Výsledkem je následující obrazovka.

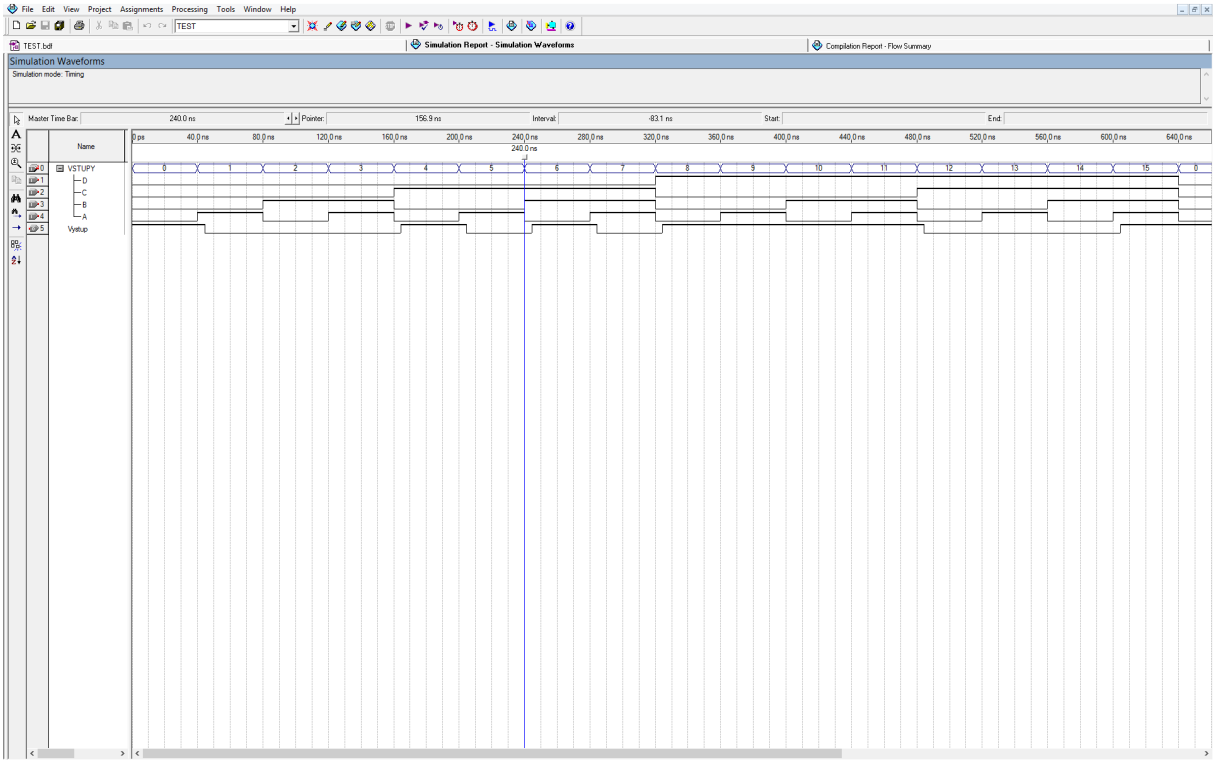


Proměnné jde složit do skupiny (VSTUPY) a její stav vyjádřit dekadicky, hexadecimálně nebo binárně viz. ukázka projektu TEST_1.



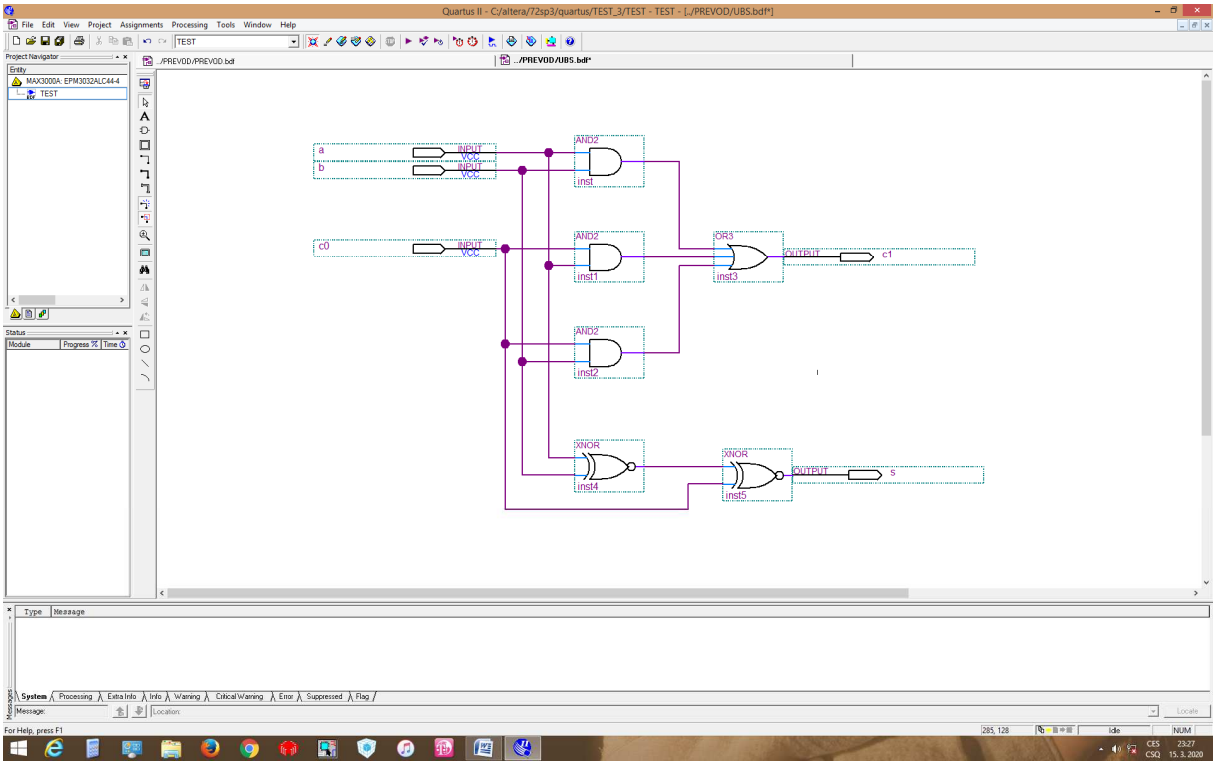
Pro práci s obvody z TTL řady budeme pod ikonou hradla otevírat položku **others - maxplus2** a najdeme příslušný obvod např. dekodér **74138** nebo multiplexer **74151** viz. ukázka projektu TEST_2 a následující dva obrázky. Symbol uzemění (GND) a logické jedničky (VCC) najdeme pod ikonou hradla **primitives - other.**



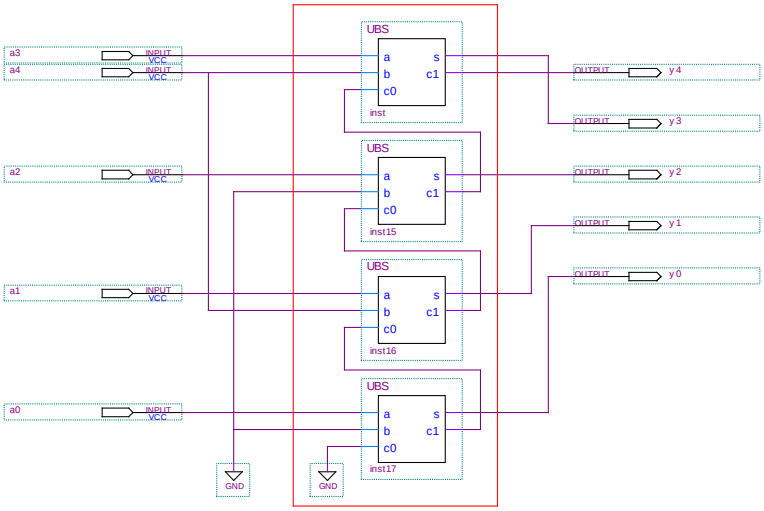


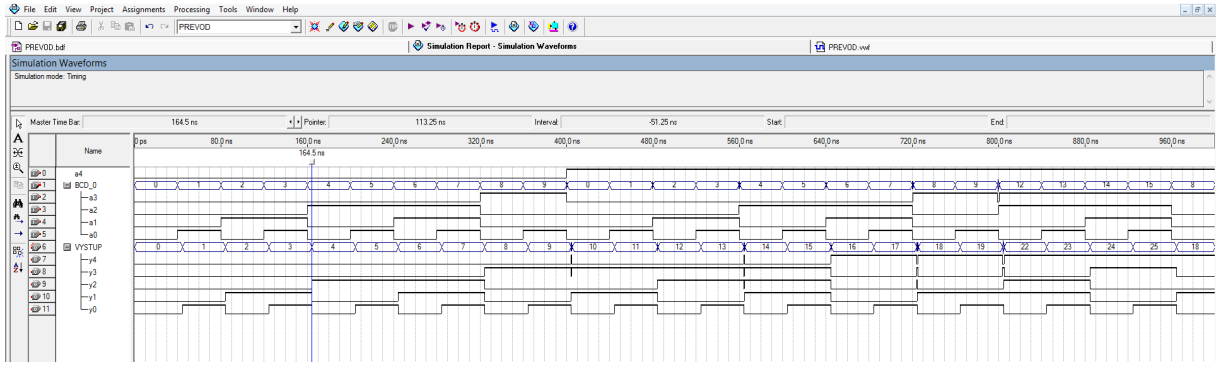
Další možnosti realizace logických obvodů v prostředí QuartusII.

Máme-li složitější schéma nebo potřebujeme v realizaci opakovat stejnou část obvodu, je potřeba pracovat s více schématy nebo stejnými bloky. Při vytváření projektu realizovaného jedním schématem nebo souborem VHDL se bude jméno projektu shodovat s hlavním projektovým souborem. V případě, kdy se projekt skládá z více dílčích částí, potom musí být jméno projektu shodné s TOP souborem, který v sobě obsahuje odkazy na jednotlivé dílčí části a jejich propojení. Příkladem může být například realizace příkladu ze cvičení 4 příklad 2. Pětibitové číslo v BCD kódu (a_4 $a_3a_2a_1a_0$) vyjadřuje číslo takto $10*a_4 + a_3a_2a_1a_0$. K realizaci budeme potřebovat jednu 4bitovou sčítačku např. s kaskádním přenosem. Základním kamenem bude obvod UBS (úplná binární sčítačka), který realizuje součet dvou binárních čísel v jednom řádu. Tedy LKO se třemi vstupy a_i, b_i, c_i a dvěma výstupy S_i a c_{i+1}. Vytvoříme si nejprve tento logický obvod a uložíme jej pod názvem UBS. Následně z něj potřebujeme vytvořit symbol, který budeme používat ve vyšším schématu jako blok. Zmáčkneme **File - Create/_Update - Create Symbol Files for Carrent File** a uložíme pod názvem **USB.bsf**.

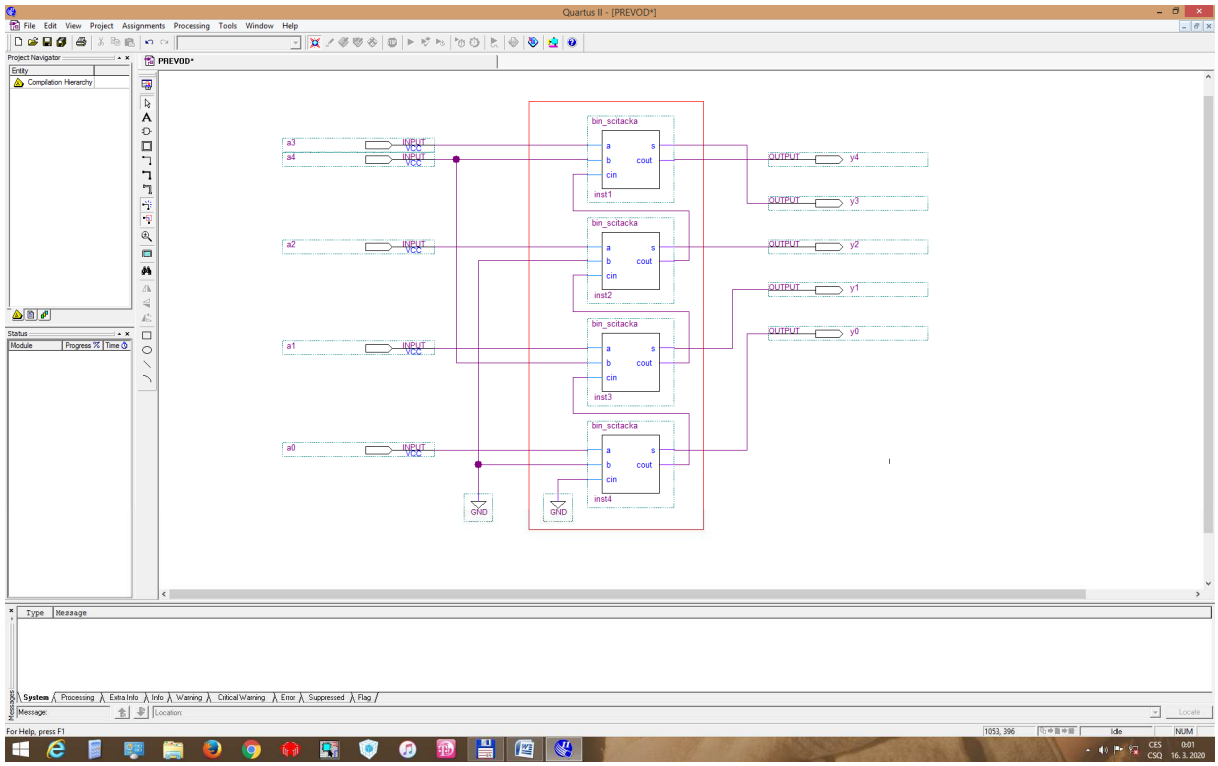


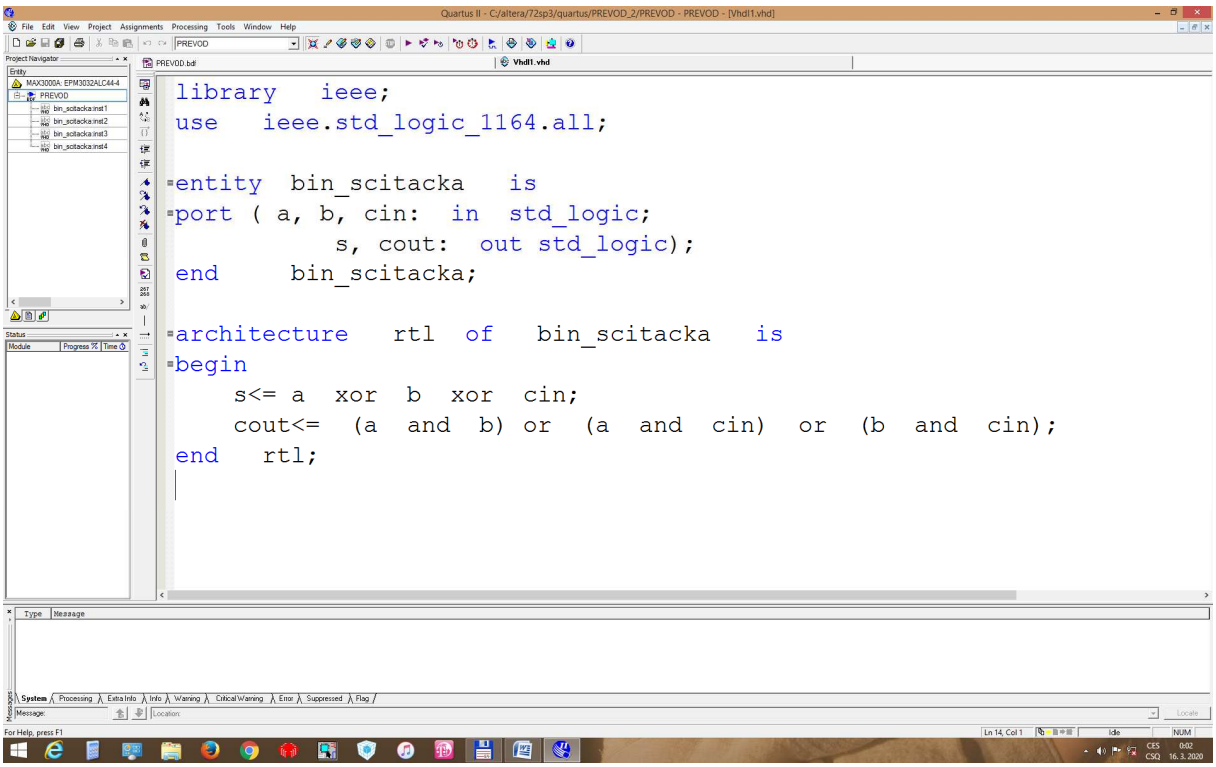
Otevřeme nový soubor, který bude **TOP** schématem a uložíme jej pod názvem projektu PREVOD. Pod ikonou hradla se objeví položka **Projekt** a pod ní naše USB. Daný blok umístíme čtyřikrát na plochu a vytvoříme kaskádní řešení čtyř bitové sčítačky (ohraničené červenou čárou) viz. následující obrázek nebo by z ní mohl být vytvořen další blok (4 bitová sčítačka). Následně si vytvoříme simulační soubor a ověříme činnost navrženého obvodu.



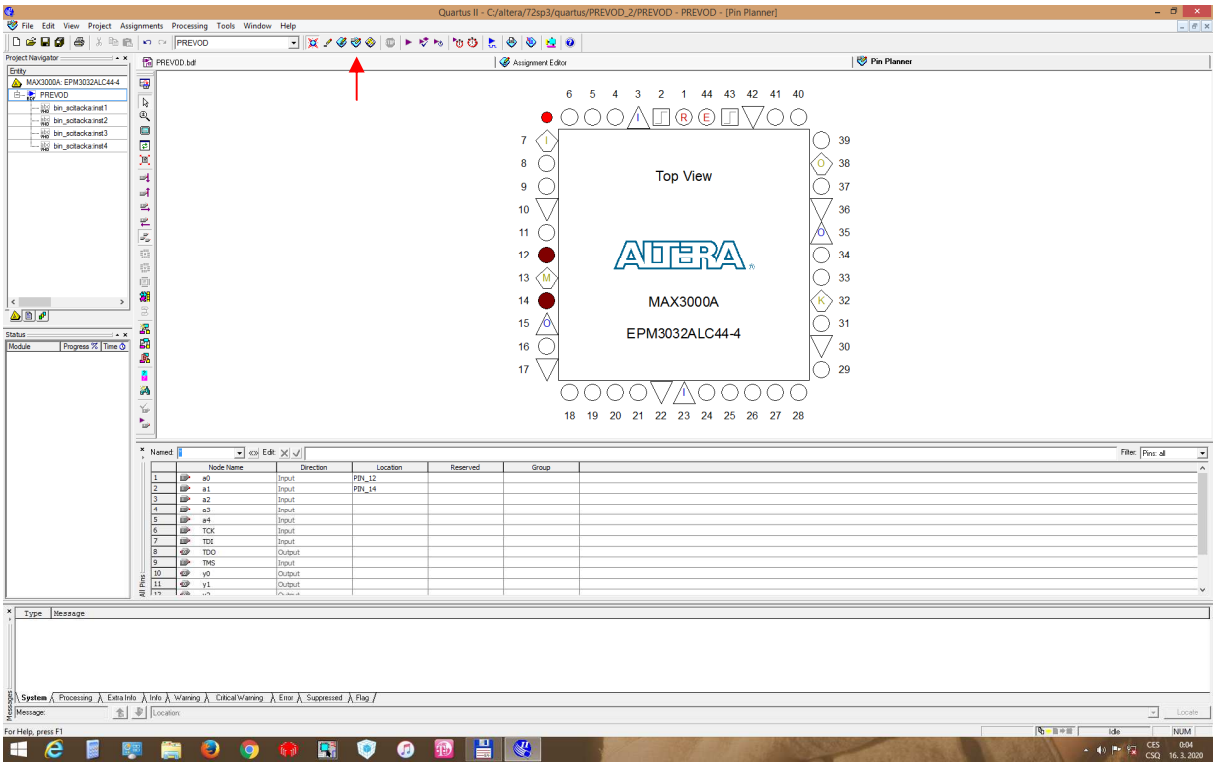


Vytvořený blok nemusí být jenom schéma, ale může být vytvořen ve firemním HDL jazyce (AHDL) nebo ve Verilogu nebo VHDL. V příkladu PREVOD_2 je binární sčítačka realizována zápisem v jazyce VHDL v tzv. modelu RTL, který popisuje rovnicemi realizaci logického obvodu. Na následujícím obrázku je zobrazen tento projekt a poklepáním na blok se nám zobrazí schéma UBS (PROJEKT) nebo zápis v jazyce VHDL (PROJEKT_2).

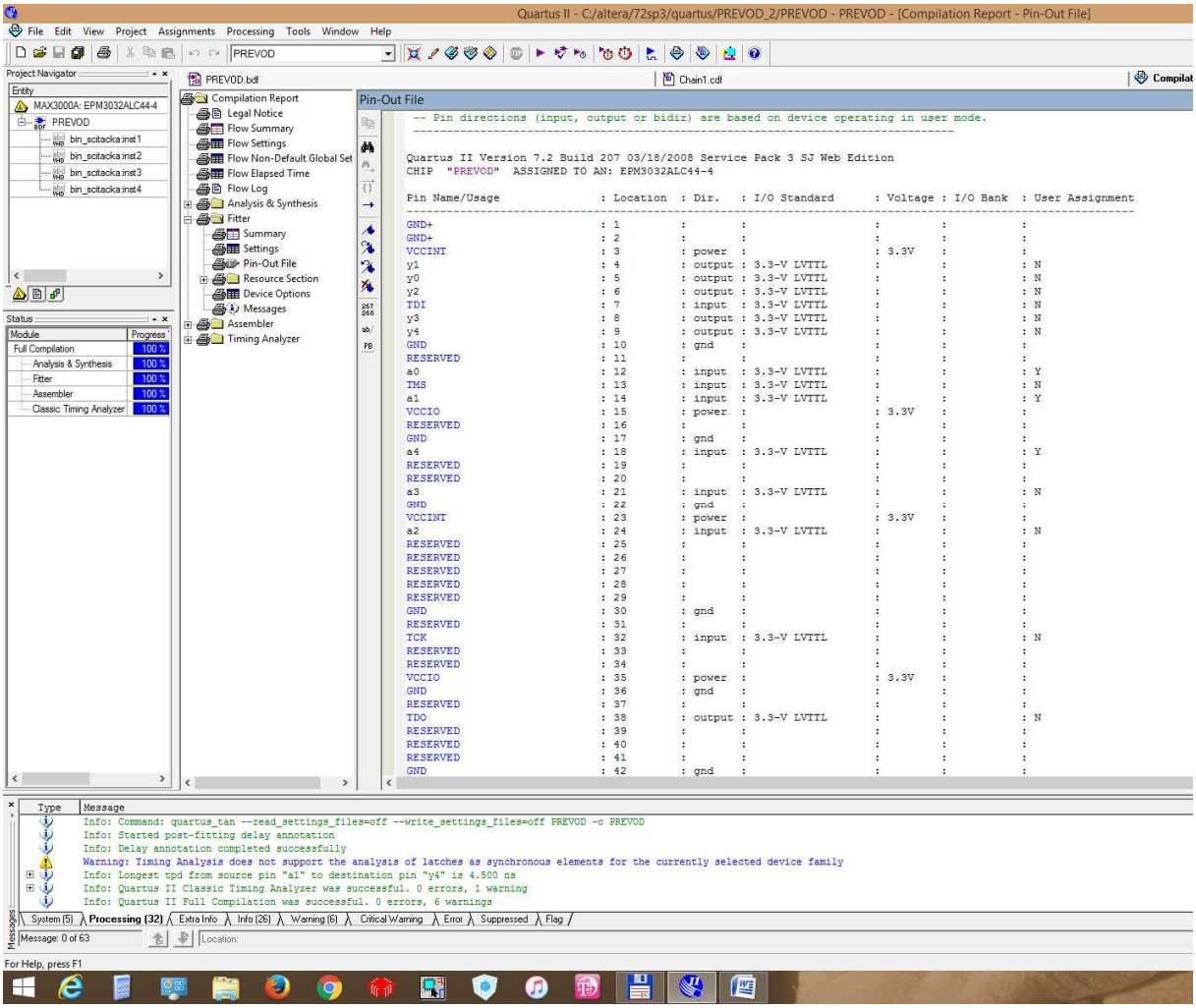




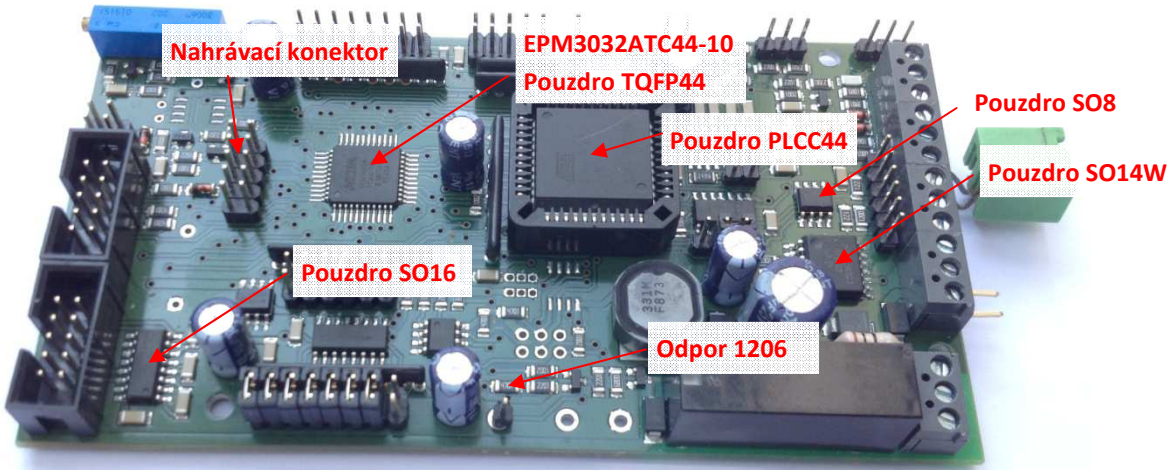
Ke skutečné realizaci obvodu musíme vyjma různých nastavení, která ze začátku pro nás nejsou důležitá, je potřeba zjistit rozmístění vývodů na čipu systémem nebo naše přiřazení proměnných jednotlivým vývodů obvodu pomocí ikony **Pin Planner** (označená šipkou).



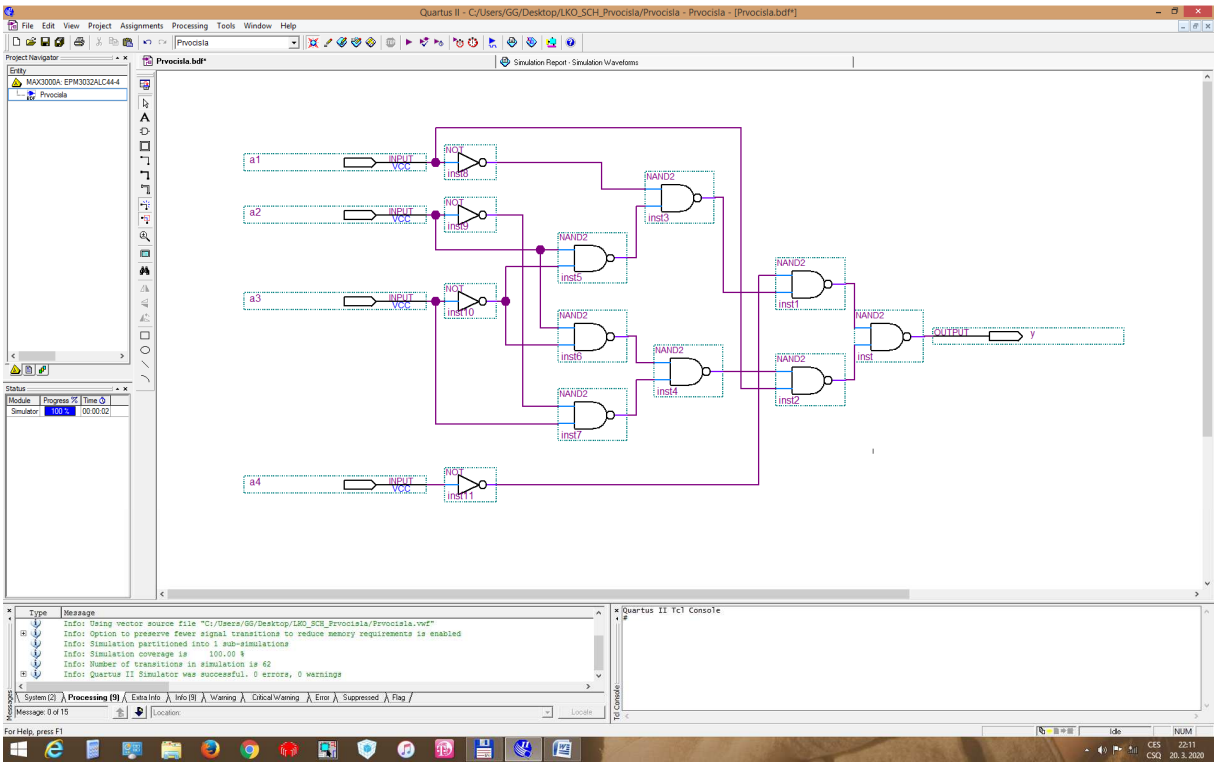
Přiřazení provedeme poklepáním na okénko **Location** a vybereme volný I/O vývod nebo ťukneme na vývod obvodu a v okénku **Pin Properties** přiřadíme proměnnou a určíme elektrické vlastnosti vývodu. Po přiřazení musíme projekt znovu přeložit a zjistit, zda jej systém zkompiluje. Pokud se rozhodneme ponechat přiřazení na systému, přiřazení vývodů najdeme ve sloupci **Compilation Report - Filter - Pin-Out File** viz.obrázek.



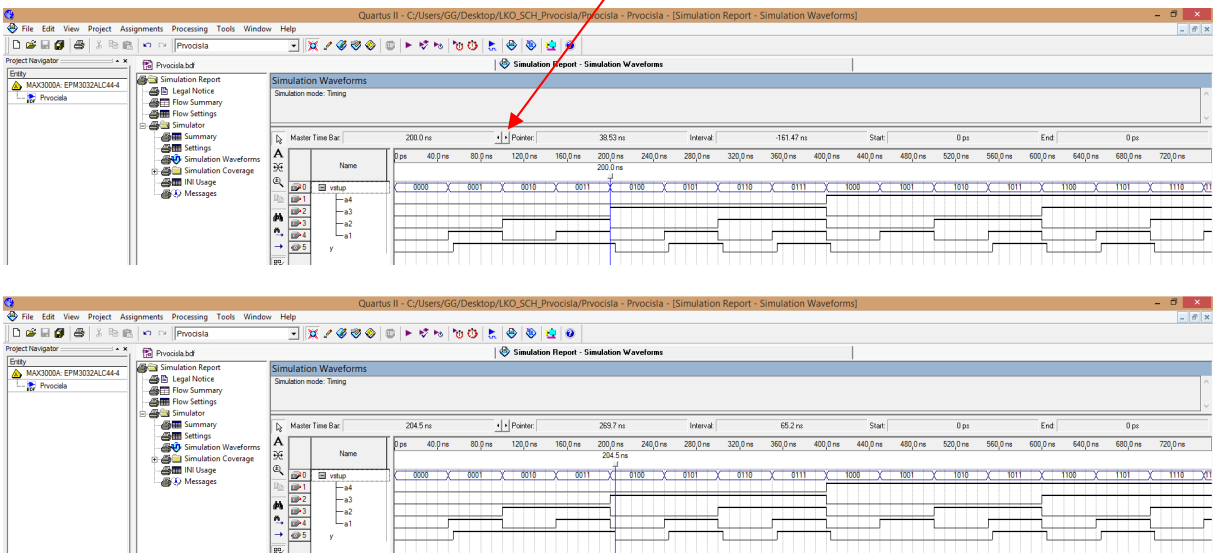
Už cca 20 let jsou všechny logické programovatelné obvody programovatelné i na osazené desce (viz. obrázek staré desky) s plošnými spoji. Posledním krokem návrhu by bylo připojení programovacího kabelu k nahrávacímu konektoru a po zmáčknutí ikony **Programmer** vybrali správný soubor. Nakonec spustili vlastní programování. U některých programovatelných obvodů je možné přes rozhraní JTAG propojit více obvodů sériově a naprogramovat je všechny najednou. Pokud obvody nejsou stejné není výběr souborů obvykle problémem. Pokud je na desce kaskádně propojeno více stejných obvodů, každý s jiným obsahem, musíme přesně vědět jak jsou za sebou zapojené a který soubor patří do kterého obvodu. **To že obvod má stejné označení neznamená, že realizuje stejnou funkci.**



Na příkladu indikátoru prvočísel si ukážeme další funkce vývojového prostředí, které jsou potřebné nebo se mohou v některých případech hodit. Nejprve si ukážeme vlastnosti na realizaci schématem. Mějme následující realizaci obvodu s obvody NAND se dvěma vstupy.

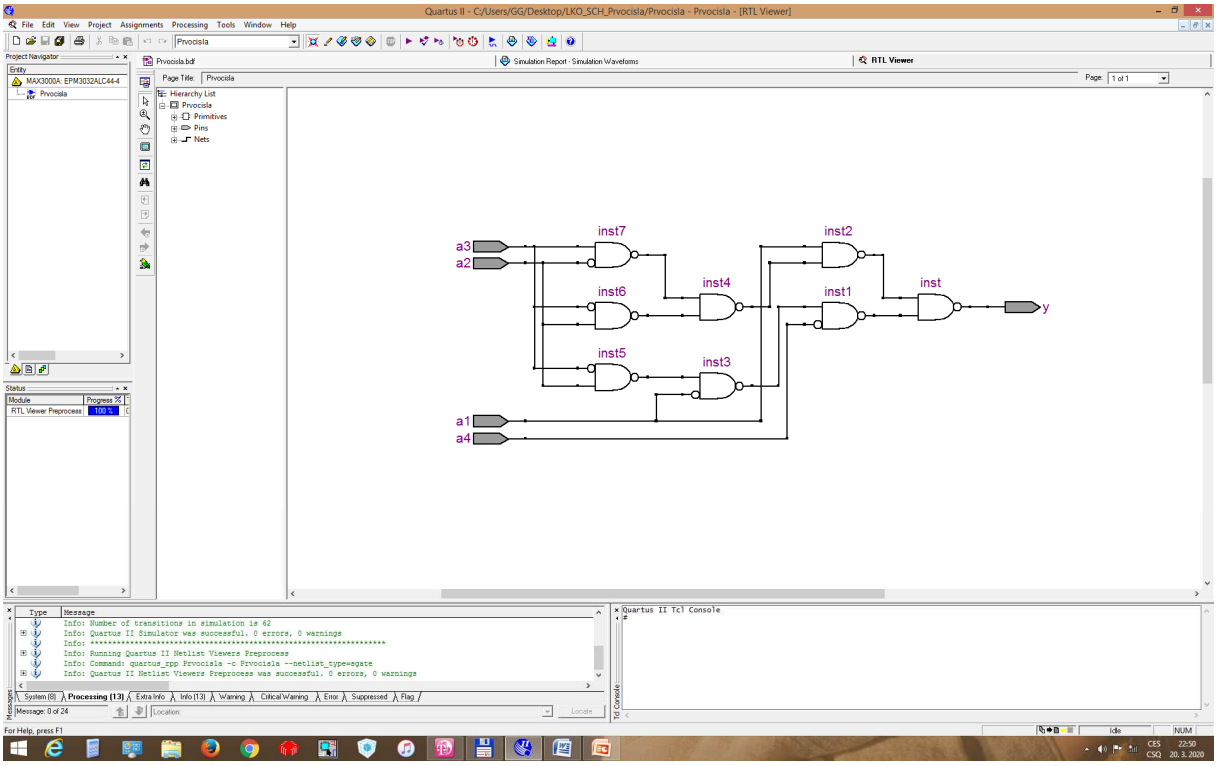


Simulací ověříme funkčnost obvodu. Nyní kliknutím na šipky Pointer můžeme posunovat modrou čáru od změny ke změně. Na prvém obrázku ukazuje na změnu vstupních proměnných (200ns) a na druhém změnu výstupu (204,5ns) pro obvod se zpožděním makrobuňky 4ns. Pro začátek je proto vhodné volit změny vstupních proměnných dostatečně velké vůči zpoždění makrobuňky obvodu. Nebude pak problém dobře vidět funkčnost obvodu a zároveň vidět, že výstup se nezmění okamžitě.

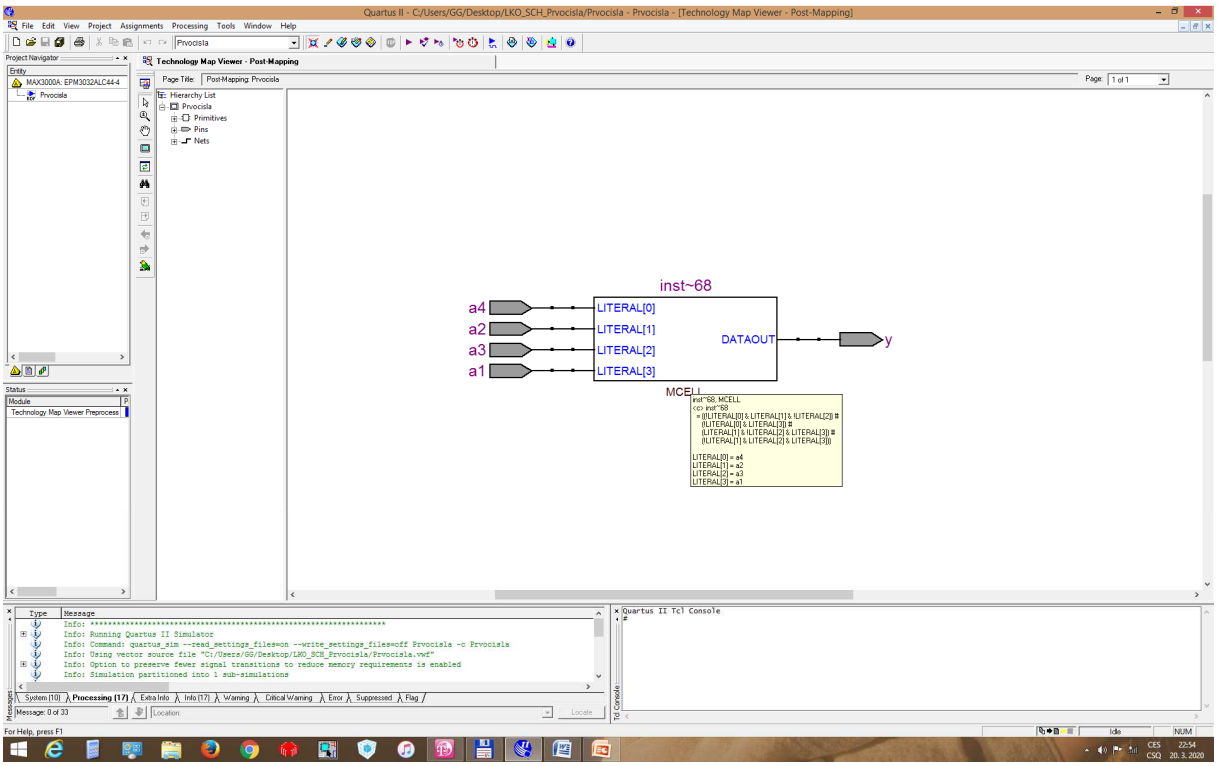


Druhá užitečná funkce (zvláště při realizaci obvodů v behaviorálním zápisu v jazyce VHDL) nám umožňuje prohlédnout si jak je obvod systémem realizován. Pro nás jsou nejvýznamnější položky **Tools - Netlist Viewers - RTL Viewer** a **Tools - Netlist Viewers - Technology Map Viewer**. První položka nám umožňuje nahlédnout na realizaci obvodu. Druhá položka nám umožňuje prohlédnout

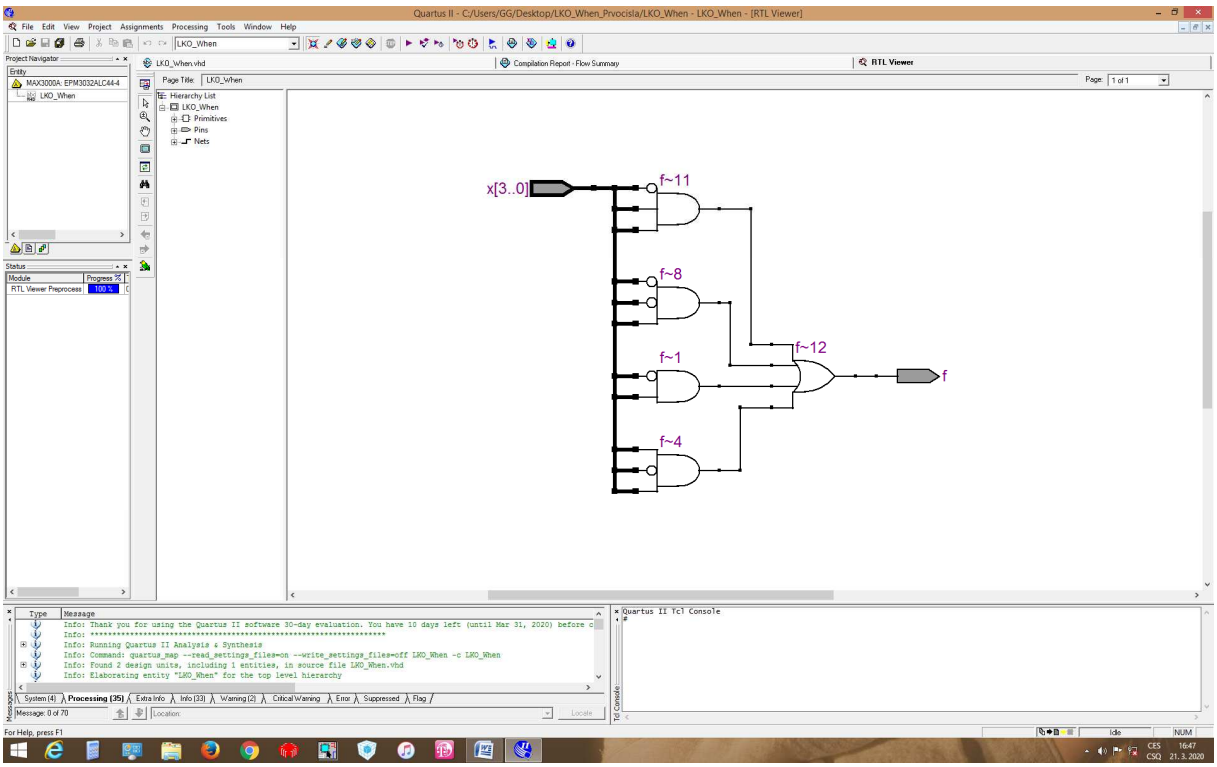
logické rovnice realizované v jednotlivých blocích nebo makrobuňkách. Pro obvod popsaný schématem bude situace vypadat následovně pro RTL Viewer.



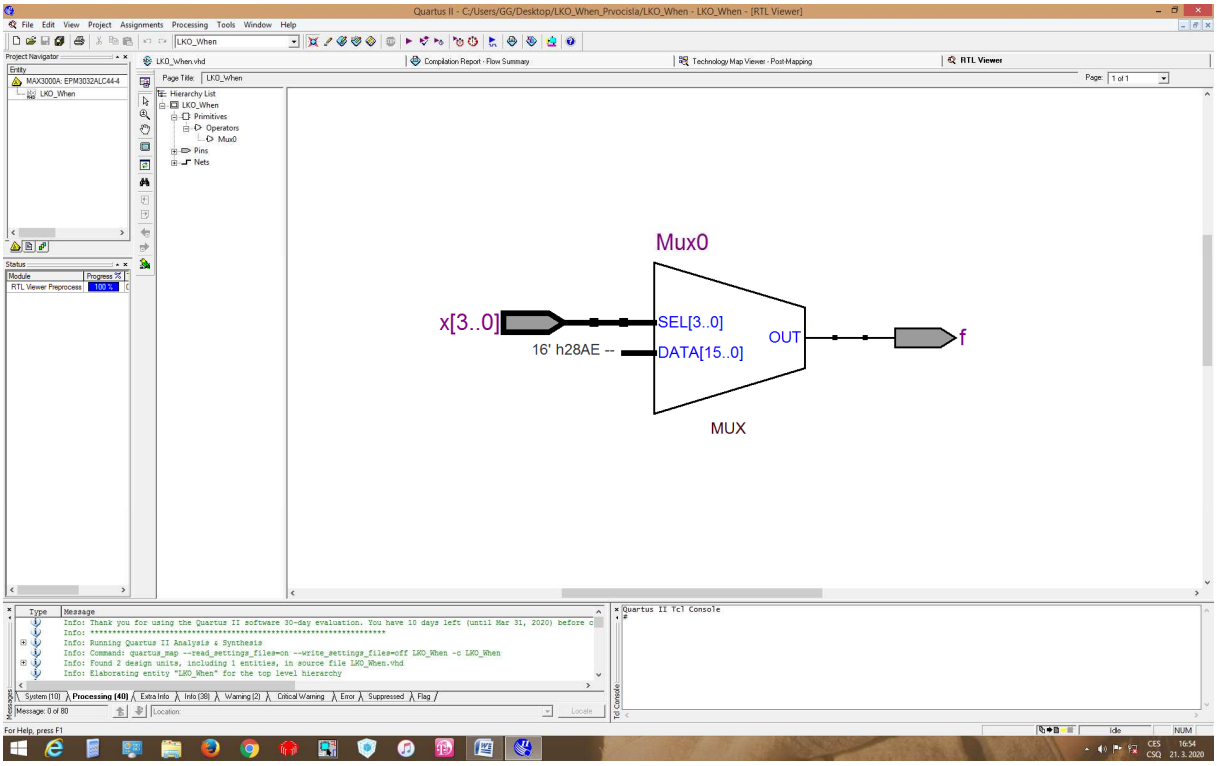
V položce Technology Map Viewer můžeme po ukázání na blok a počkání nebo stiskem levého tlačítka na myši vidět okno s logickými rovnicemi.



Bude-li obvod popsán ve VHDL tak se situace změní následně. Jestliže obvod popíšeme rovnicemi (první možný zápis v projektu LKO_When), pak RTL Viewer nám ukáže schéma a Technology Map Viewer rovnice jako na předcházejícím obrázku.

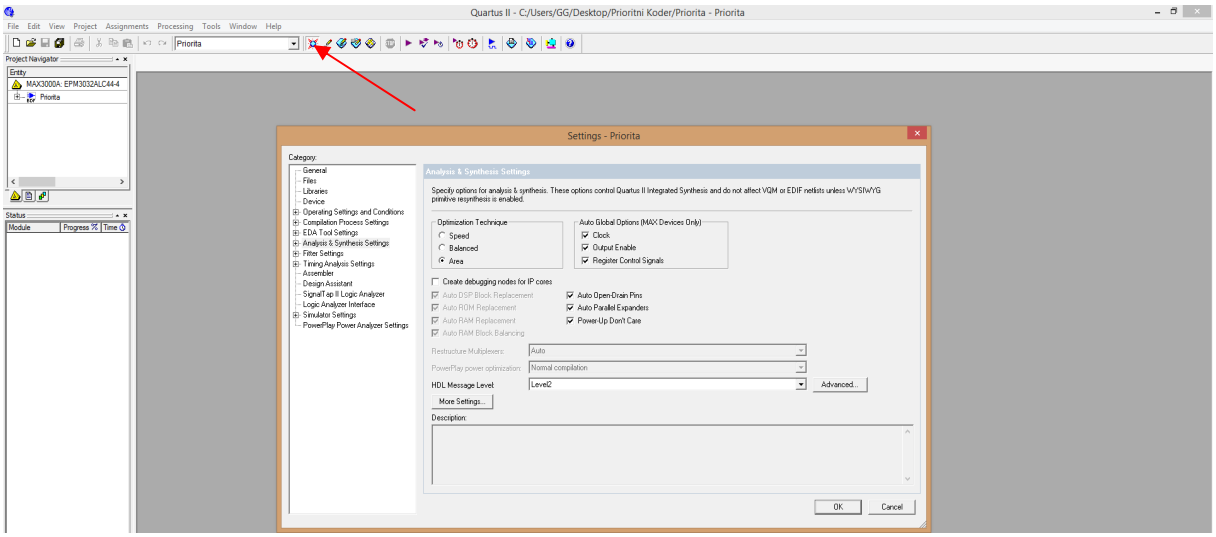


Jestliže obvod popíšeme behaviorálním zápisem (druhý, třetí nebo čtvrtý možný zápis v projektu LKO_When), pak RTL Viewer nám ukáže schematické řešení (často využívající multiplexer nebo paměť v závislosti na typu programovatelného obvodu) a Technology Map Viewer ukáže rovnice realizované v daném bloku.

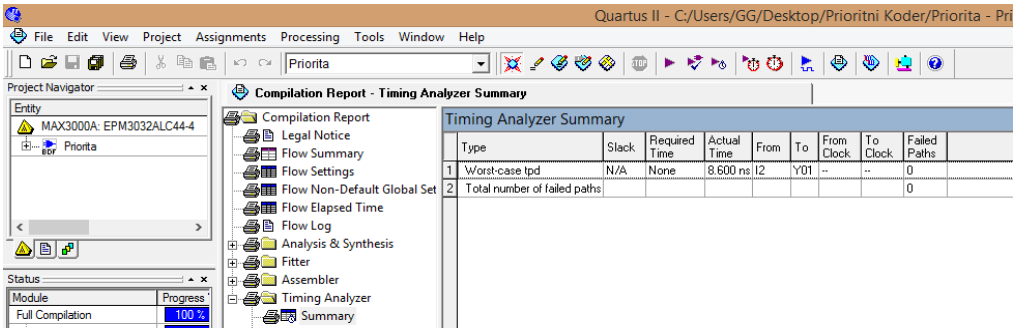


Obrázek nám říká použití multiplexeru 1 z 16, kde na jeho vstupy jsou přivedené funkční hodnoty pro jednotlivé vstupní stavy. Hodnota 28AEh = 0010100010101110 tj. stavy výstupu pro kombinace 15 až 0.

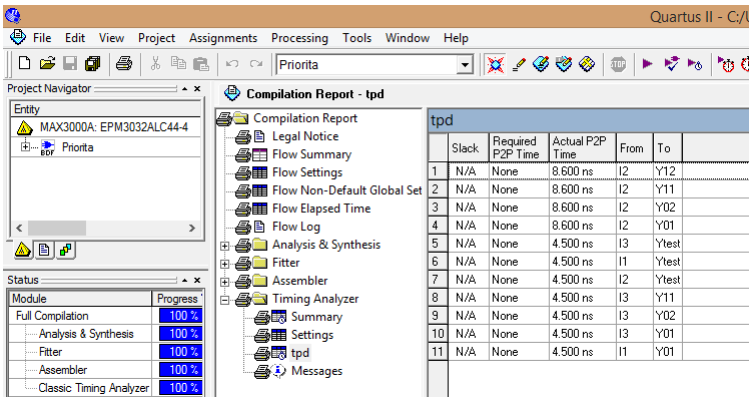
Třetí užitečnou funkcí je určení, zda Váš návrh má být realizován na co nejvyšší rychlost, co nejmenší prostor (obsazení jednotlivých buněk) nebo má být vyvážený mezi těmito krajními případy. Rozdíly mezi těmito nastaveními se ale budou projevovat až u větších návrhů. Nastavení najdeme pod položkou **Assignment - Settings - Analysis & Synthesis Settings - Optimization Technique** viz. obrázek.



Poslední potřebnou funkcí pro reálný návrh je zjištění největšího zpoždění mezi vstupem a výstupy, případně všech zpoždění mezi vstupy a výstupy. Máme-li přeložený projekt, pak v **Compilation Report - Flow Summary** (okénko se po překladu otevře samo nebo jej získáme stiskem ikony na kterou ukazuje šipka na předcházejícím obrázku), je položka **Timing Analyzer**. Pod položkou **Summary** získáme informaci o největším zpoždění v navrženém obvodu.

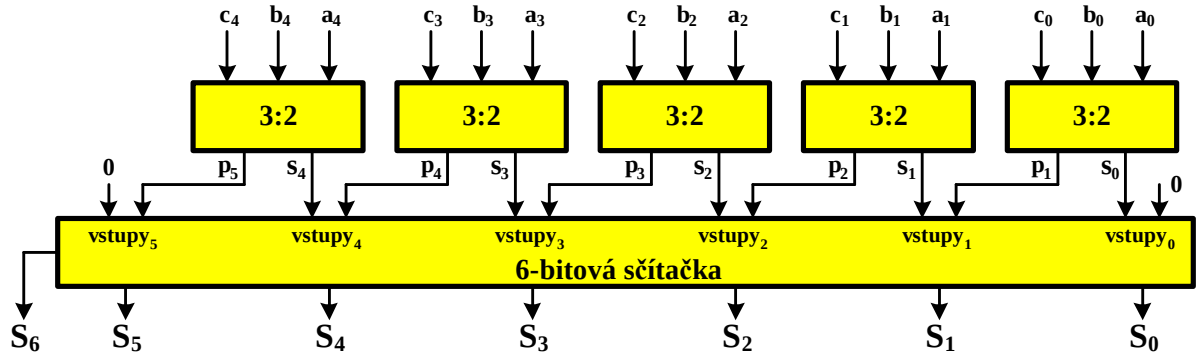


Pod položkou **tpd** získáme přehled všech zpoždění mezi vstupy a výstupy navrženého obvodu.



Pokud budeme realizovat ve VHDL složitější logický obvod, který je složen z několika stejných nebo různých částí (bloků), je potřeba si ukázat jak můžeme strukturovat takový návrh. V realizaci se bude objevovat položka **component** (obdoba bloku nebo opakujícího se bloku použitého ve schématu viz.(binární sčítačka)). Druhou položkou bude **signal,** což je proměnná použitá k vnitřnímu propojení bloků. V direktoráři takového návrhu musí být umístěn nejen soubor s **TOP** návrhem, ale i VHDL soubory se všemi návrhy použitých dílčích obvodů (commponent). Jako ukázku si ukážeme součet tří pětibitových čísel využívající urychlující techniku tzv. kompresorů (v našem případě 3:2, realizovaný UBS). Vycházet budeme z následujícího schématu.

	a_4	a_3	a_2	a_1	a_0
	b_4	b_3	b_2	b_1	b_0
	c_4	c_3	c_2	c_1	c_0
0	s_4	s_3	s_2	s_1	s_0
p_5	p_4	p_3	p_2	p_1	0
S_6	S_5	S_4	S_3	S_2	S_1	S_0



Vpravo je zobrazen způsob výpočtu. Pod první čarou je součet tří bitů vyjádřený součtovým bitem s_i a přenosem do dalšího řádu p_i pro i ∈<0÷5>. Šestibitová sčítačka pak provede součet součtových bitů s přenosy. Realizaci tohoto obvodu provedeme ve třech krocích. Nejprve v jazyce VHDL vytvoříme soubor realizující kompresor 3:2, který je shodný s úplnou binární sčítačkou. Potom vytvoříme soubor pro realizaci 6-ti bitové sčítačky s kaskádním přenosem, která bude obsahovat komponentu UBS. Nakonec vytvoříme soubor realizující sčítačku tří pětibitových čísel obsahující komponentu UBS a 6-ti bitové sčítačky.

Prvním souborem bude **bin_scitacka.vhd,** s kterým jsme se již setkali.

```
library  ieee;
use  ieee.std_logic_1164.all;

entity bin_scitacka  is
port ( a, b, cin:  in  std_logic;
       s, cout:  out std_logic);
end    bin_scitacka;

architecture  rtl  of  bin_scitacka  is
begin
  s<= a  xor  b  xor  cin;
  cout<= (a  and  b) or  (a  and  cin)  or  (b  and  cin);
end  rtl;
```

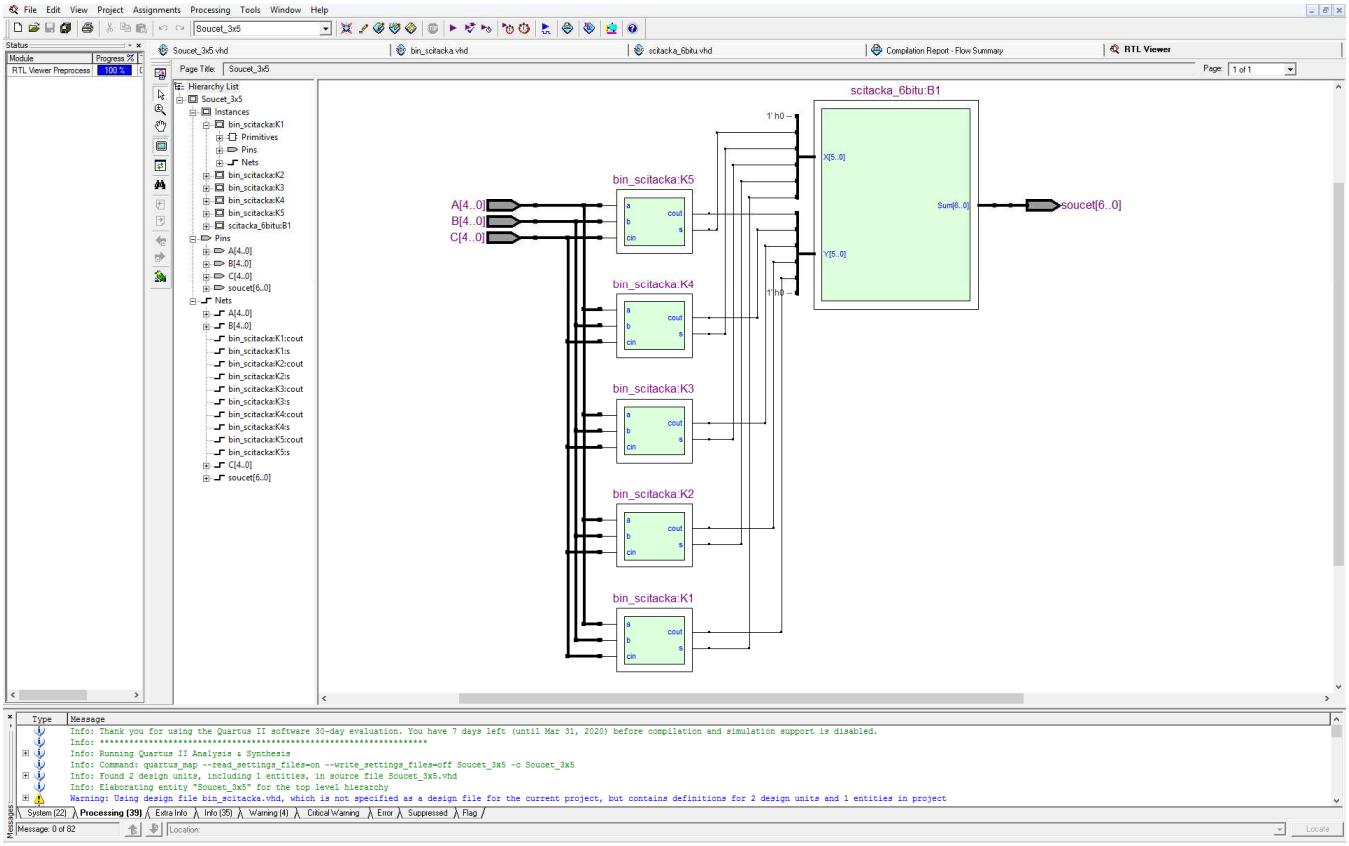
Druhým souborem bude **scitacka_6bitu.vhd**, který využívat komponentu bin_scitacka, jejíž zápis ve VHDL je uveden výše. sčítačka má 2x6 vstupů pro čísla označená X a Y. Výstup může být až 7-mi bitový a je označen Sum. V jeho těle architecture musí být vložena definice commponent bin_scitackaje s uvedením vstupů a výstupů, kterými disponuje. tato komponenta potřebuje propojit kaskádní přenos mezi jednotlivými UBS a k tomu jsou definovány signály p1 až p6. Nakonec je realizováno vlastní propojení mezi sčítačkami s identifikací daného bloku (B1: až B6:). V označení je libovolné vyjma vyhrazených slov a v návrhu nesmí být dva bloky označeny stejně. V některých případech může být počet bloků značný a jejich výpis zdlouhavý. V některých případech je možné opakující se bloky umístit do cyklu, jak je ukázáno s TOP návrhu **Soucet_3x5.vhd**.

```
library ieee;
use  ieee.std_logic_1164.all;

entity scitacka_6bitu  is
port (X, Y: in  std_logic_vector (5 downto 0);
     p0:   in  std_logic;
     Sum:  out  std_logic_vector (6 downto 0));
end  scitacka_6bitu;

architecture  structural  of  scitacka_6bitu is
component  bin_scitacka   is
port ( a, b, cin:  in  std_logic;
      s, cout:  out std_logic);
end component;

signal  p1, p2, p3, p4, p5, p6:  std_logic;   --vytvoření šesti vnitřních signálů představující
begin                                          --přenosy mezi binárními sčítačkami.
B1: bin_scitacka   port map (a=> X(0), b=>Y(0), cin=>'0', s=>Sum(0) , cout=>p1);
B2: bin_scitacka   port map (a=> X(1), b=>Y(1), cin=>p1, s=>Sum(1) , cout=>p2);
B3: bin_scitacka   port map (a=> X(2), b=>Y(2), cin=>p2, s=>Sum(2) , cout=>p3);
B4: bin_scitacka   port map (a=> X(3), b=>Y(3), cin=>p3, s=>Sum(3) , cout=>p4);
B5: bin_scitacka   port map (a=> X(4), b=>Y(4), cin=>p4, s=>Sum(4) , cout=>p5);
B6: bin_scitacka   port map (a=> X(5), b=>Y(5), cin=>p5, s=>Sum(5) , cout=>Sum(6));
end   structural;

-- ZAPIS      port map (a=> X(0), b=>Y(0), cin=>'0', s=>Sum(0) , cout=>p1);
-- BY MOHL BYT ZAPSAN     port map (X(0),Y(0),'0',Sum(0) ,p1);
-- V prvním zápise je zřejmé propojení vývodů, v druhém můžeme snadno chybovat.
```

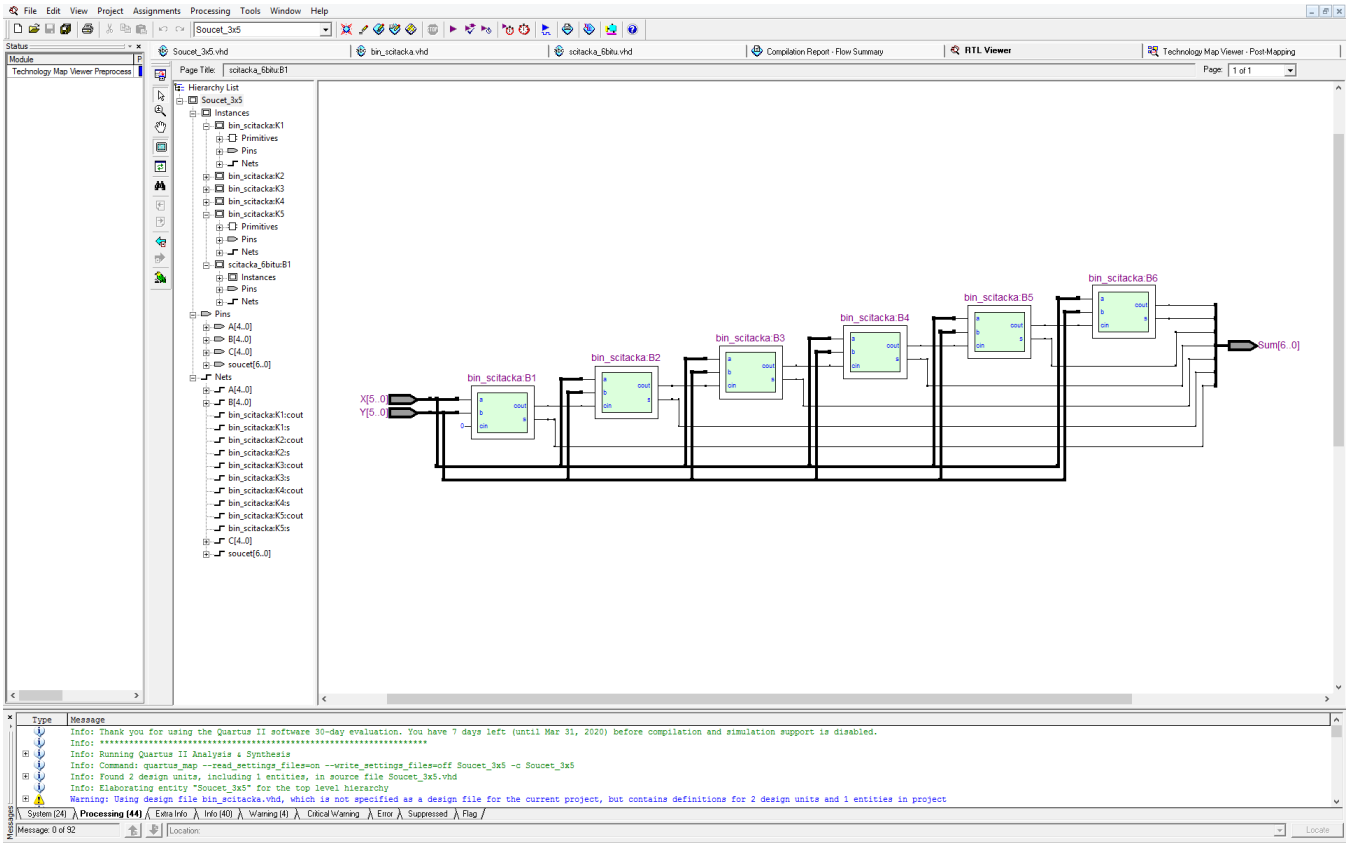
Posledním souborem bude TOP návrh realizovaný souborem **Soucet_3x5.vhd**.

```
library ieee;
use  ieee.std_logic_1164.all;

entity Soucet_3x5 is
generic (length:  integer:=5);
port (A, B, C: in  std_logic_vector (length-1 downto 0);
    soucet:  out  std_logic_vector (length+1 downto 0));
end  Soucet_3x5;

architecture  structural  of  Soucet_3x5 is

component  bin_scitacka   is
port ( a, b, cin:  in  std_logic;
      s, cout:  out std_logic);
end component;
component  scitacka_6bitu   is
port (X, Y: in  std_logic_vector (5 downto 0);
    Sum:    out  std_logic_vector (6 downto 0));
end component;

signal so: std_logic_vector (5 downto 0);  --vytvoření šesti vnitříních signálů představující
                                           --součty z kompresorů 3:2
signal p: std_logic_vector (5 downto 0);   --vytvoření šesti vnitřních signálů představující
                                           --přenosy z kompresorů 3:2
begin              --KOMPRESE BITU
K1: bin_scitacka   port map (a=> A(0), b=>B(0), cin=>C(0), s=>so(0) , cout=>p(1));
K2: bin_scitacka   port map (a=> A(1), b=>B(1), cin=>C(1), s=>so(1) , cout=>p(2));
K3: bin_scitacka   port map (a=> A(2), b=>B(2), cin=>C(2), s=>so(2) , cout=>p(3));
K4: bin_scitacka   port map (a=> A(3), b=>B(3), cin=>C(3), s=>so(3) , cout=>p(4));
K5: bin_scitacka   port map (a=> A(4), b=>B(4), cin=>C(4), s=>so(4) , cout=>p(5));

--nebo kratším zápisem (né vždy je tak jednoduše realizovatelný)
--komprese: for i in 0 to 4 generate
--K1: bin_scitacka  port map (a=> A(i), b=>B(i), cin=>C(i), s=>so(i) , cout=>p(i+1));
--        end generate;

                -- SOUCET dílčích součtů a přenosů
p(0)<='0'; so(5)<='0';
B1: scitacka_6bitu   port map (X=> so(5 downto 0), Y=> p(5 downto 0), Sum=> soucet(6 downto 0));
end   structural;
```

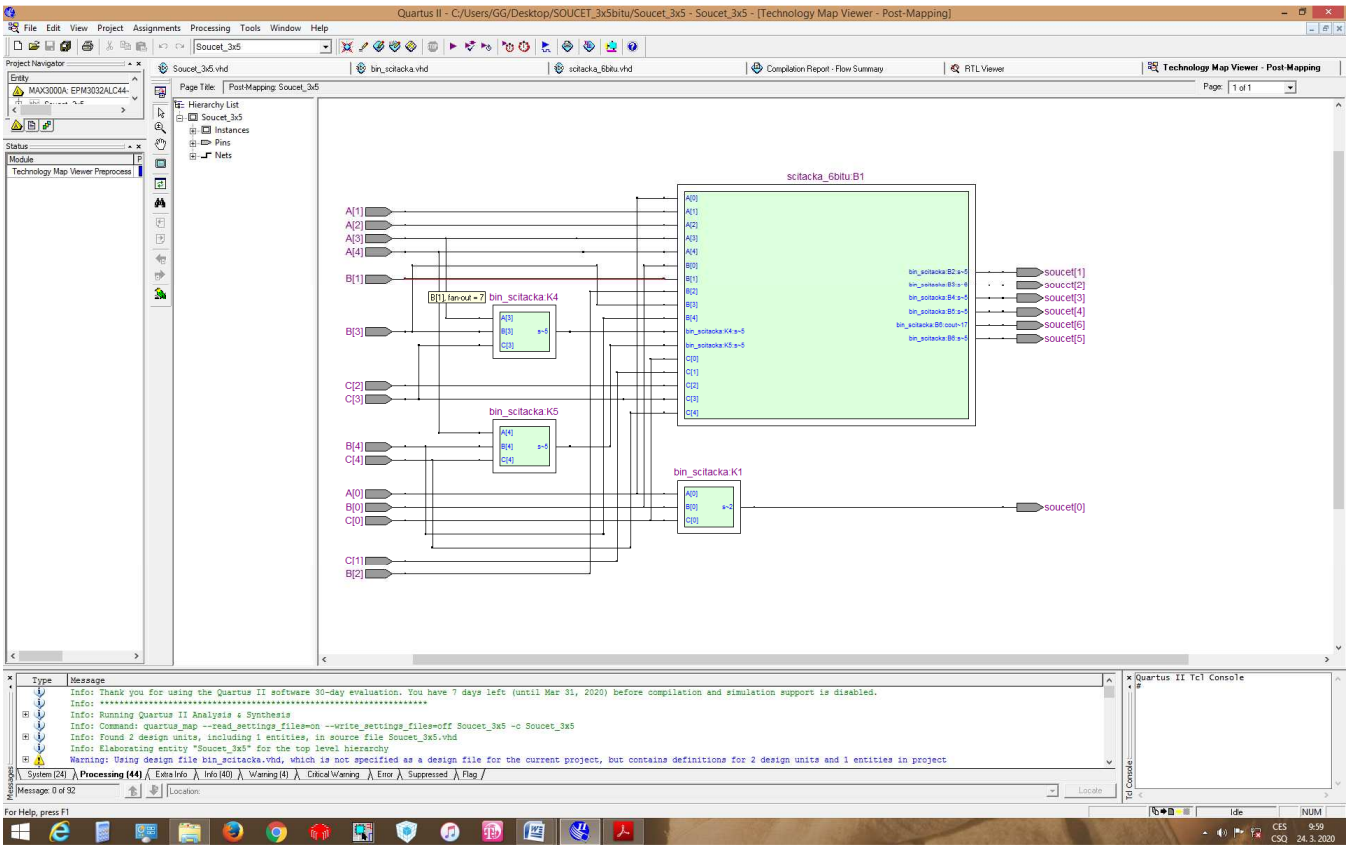
Celý projekt je uložen v direktoráři Soucet_3x5. Na obrázku je zobrazen výstup RTL Viewer.



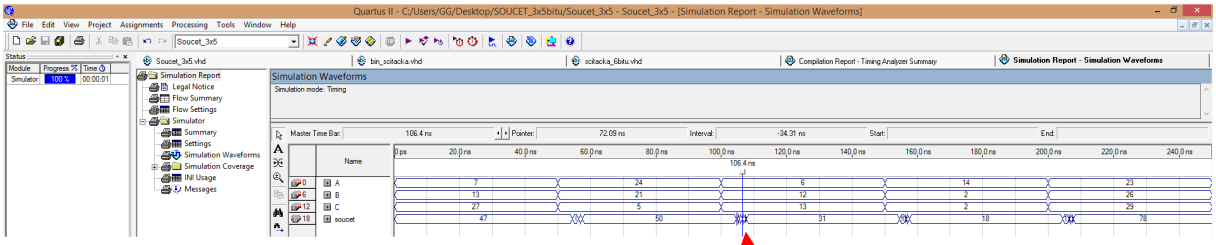
Rozevřením bloku sčítačka_6bitů:B1 přes příkaz **Hierarchy Down** uvidíme propojení jednotlivých bloků USB.



V položce **Tools - Netlist Viewers - Technology Map Viewer**, pak můžeme vidět celou strukturu, jak byla zrealizována systémem.



Přes příkaz **Hierarchy Down** se můžeme dostat až na rovnice realizovaných bloků. V simulaci ověříme chování navrženého obvod, který má největší zpoždění pro EPM3032ALC44-4 8,6ns.



Při rozšíření časové osy vidíme, že změna výstupní hodnoty prochází přes několik stavů, než se ustálí. Pro použitelnou realizaci bychom potřebovali do struktury přidat paměťové členy, do kterých ukládat vstupní čísla a s příslušným zpožděním uložit stabilní hodnotu součtu.