VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Použití časových omezení při návrhu programovatelné logiky – laboratorní úloha.

Using of Timing Constraints in Programmable Logic Design - Laboratory Exercise.

2015 Jan Nožička

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Jan Nožička**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Použití časových omezení při návrhu programovatelné logiky – laboratorní úloha
Using of Timing Constraints in Programmable Logic Design - Laboratory Exercise

Zásady pro vypracování:

1. Seznámení se s technikou FPGA a návrhem programovatelných logických obvodů.
2. Studium problematiky časových omezení při návrhu FPGA.
3. Návrh koncepce úlohy pro demonstraci vlivu časových omezení.
4. Vytvoření laboratorní úlohy s využitím návrhového prostředí Xilinx ISE.
5. Experimentální ověření úlohy na vývojové desce NEXYS-3.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL.* 1. vyd. Praha: BEN - technická literatura, 2006. 349 s. ISBN 80-7300-198-5.
[2] ŠŤASTNÝ, Jakub. *FPGA prakticky.* Praha: BEN - technická literatura, 2011. ISBN 978-80-7300-261-9.
[3] XILINX, Inc. *Constraints Guide.* [online]. [cit. 13.10.2013]. Dostupné z: http://www.xilinx.com/itp/xilinx10/books/docs/cgd/cgd.pdf.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Vladimír Kašík, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

7. 5. 2015

Jan Nožička

Abstrakt

Cílem bakalářské práce je vytvoření laboratorní úlohy pro demonstraci vlivů časových omezení při návrhu programovatelné logiky v FPGA. Stručně je popsána technika FPGA a technologie návrhu programovatelných hradlových polí pomocí jazyku VHDL. Následuje seznámení s dostupnými omezeními, kde jsou především rozebrána časová omezení, která jsou zahrnuta do laboratorní úlohy.

Výsledky této práce umožňují praktické ověření výhod časových omezení pro návrh logiky v FPGA. Výhody spočívají jednak v navýšení pracovní frekvence, úspoře použité logiky a díky kontrole parametrů hodinového signálu můžeme předejít tvorbě nejistot, které mohou ohrozit funkčnost celého obvodu.

Smyslem práce je bližší seznámení, předně studentů s návrhem a funkčností časových omezení, přičemž práce může sloužit také jako český návod k dalšímu vzdělávání v oblasti omezení obecně.

Abstract

The aim of this Bachelor Thesis is the creation of laboratory task for the demonstration of timing constraints influences during suggestion of programmable logic in FPGA. Briefly, there is a description of FPGA techniques and creation technology of field programmable gate array thanks to VHDL language in this work. Further meeting available constraints, mainly focused on timing constraints that are included into the laboratory task.

The results of this Bachelor Thesis enable practical verification of timing constraints advantages for suggestion of FPGA logic. These benefits contains the increase of working frequency, saving of used logic, and thanks to the control of clock signal parameters we can avoid the creation of uncertainty that can endanger functionality of the whole circuit.

The aim of this Bachelor Thesis is to closely describe, especially to students, suggestion and functionality of timing constraints. The work itself can also serve as Czech instructions to further education in the area of constraints generally.

Klíčová slova

FPGA; Časová omezení; VHDL; Perioda; UCF

Key words

FPGA; Timing Constraints; VHDL; Period; UCF

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Vladimíru Kašíkovi Ph.D. za odborné rady a cenné připomínky při vypracování a návrhu mé bakalářské práce.

Seznam použitých symbolů a zkratek

ASCII	tabulka kódování znaků ve výpočetní technice
ASIC	obvod naprogramován již při výrobě
CLB	konfigurační logická buňka
f	frekvence [Hz]
FPGA	čip programovatelného hradlového pole
I/O	vstupně/výstupní piny
LUT	logická buňka
NCD	soubor procesu implementace
NCF	soubor omezení logických prvků a jejich vzájemného propojení
NGD	soubor procesu implementace
ns	nanosekundy
PCF	soubor omezení fyzického umístění
ps	pikosekundy
RAM	paměť
T	perioda [s]
UCF	soubor uživatelských omezení
VHDL	jazyk pro popis hadrware

Obsah:

1. Úvod

Programovatelné logické obvody si během posledních let prošli velkým technologickým vývojem. Od vskutku jednoduchých, které plnili funkci základní kombinační logiky až po plně programovatelné s mnoha vstupy, vnitřní pamětí a pracovní frekvencí stovek MHz. Avšak u takových mohou vlivem složité programovatelné logiky vznikat nežádoucí zpoždění, hlavně na spojích mezi jednotlivými prvky. Toho se dá zamezit použitím vhodných časových omezení při tvorbě kódu, podle nějž má programovatelný logický obvod pracovat. Právě časovými omezeními se zabývá tahle práce, jejíž výstupem je laboratorní úloha.

2. FPGA a jazyk VHDL

2.1 FPGA

Obvody FPGA (*Field Programmable Gate Array*), tedy programovatelná hradlová pole, jsou speciální integrované obvody, které obsahují určité množství logických prvků propojených konfigurovatelnou maticí spojů. Jsou dosud nejmodernější podskupinou PLD (*Programmable Logic Device*) do které patří také výkonnostně a vývojově starší CPLD (*Complex Programmable Logic Device*) a SPLD (*Simple Programmable Logic Device*).

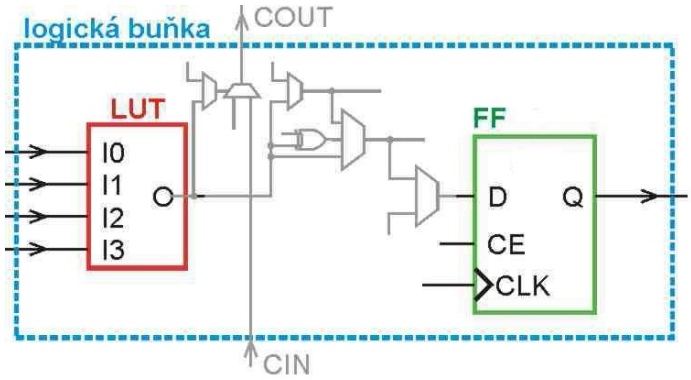
Hlavní výhody FPGA obvodů tkví určitě v jejich flexibilitě, tím je myšlena možná přeprogramovatelnost dle specifických požadavků, které se mohou měnit. Dále nesmíme opomenout paralelizaci procesů, což značně zvyšuje výkon zpracování dat, jelikož např. ukládání do paměti, nebo komunikace s periferiemi obvodu může probíhat současně.

Z toho plynoucí využití např. pro výpočetně náročné algoritmy, pro zpracování velkých datových toků, při návrhu zařízení obvodu ASIC, jako platforma pro vyzkoušení prototypu. [3],[5],[9]

2.1.1 Vnitřní struktura FPGA

Typická struktura programovatelného logického obvodu (obr. č. 2) obsahuje především tyto časti:

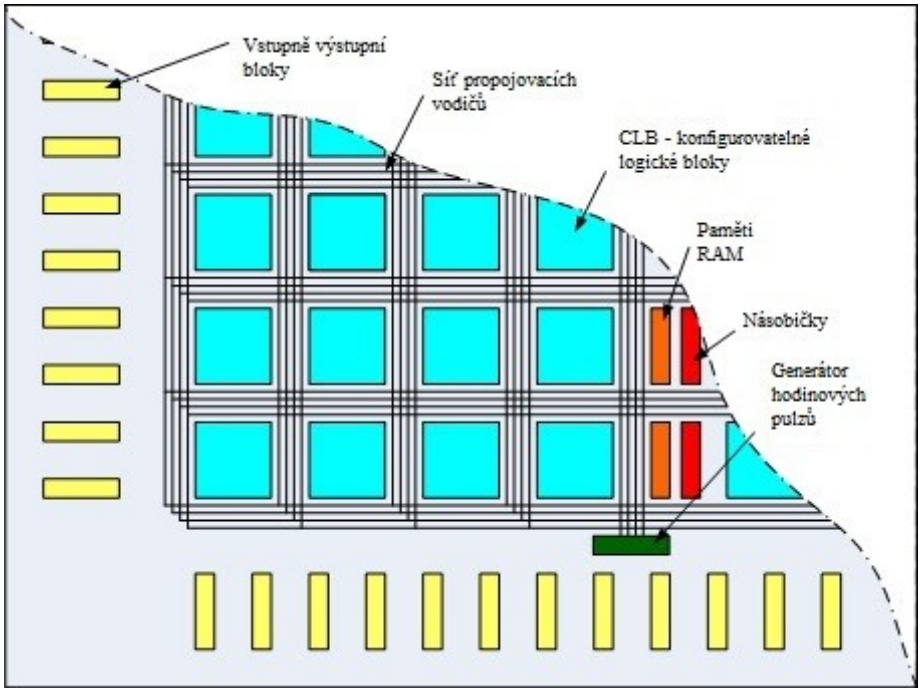
- Vstupně výstupní konfigurovatelné I/O (*Input/Output*) bloky – tvoří rozhraní mezi vnějšími vývody součástky
- Síť propojovacích vodičů - slouží k propojení logických bloků a I/O bloků
- CLB (*Configurable Logic Block*) konfigurovatelný logický blok – základní stavební prvek pro vytváření logických funkcí, obsahuje logické buňky (slices) na obr. č. 1, které se skládají z:



Obrázek 1 – zjednodušené schéma logické buňky [4]

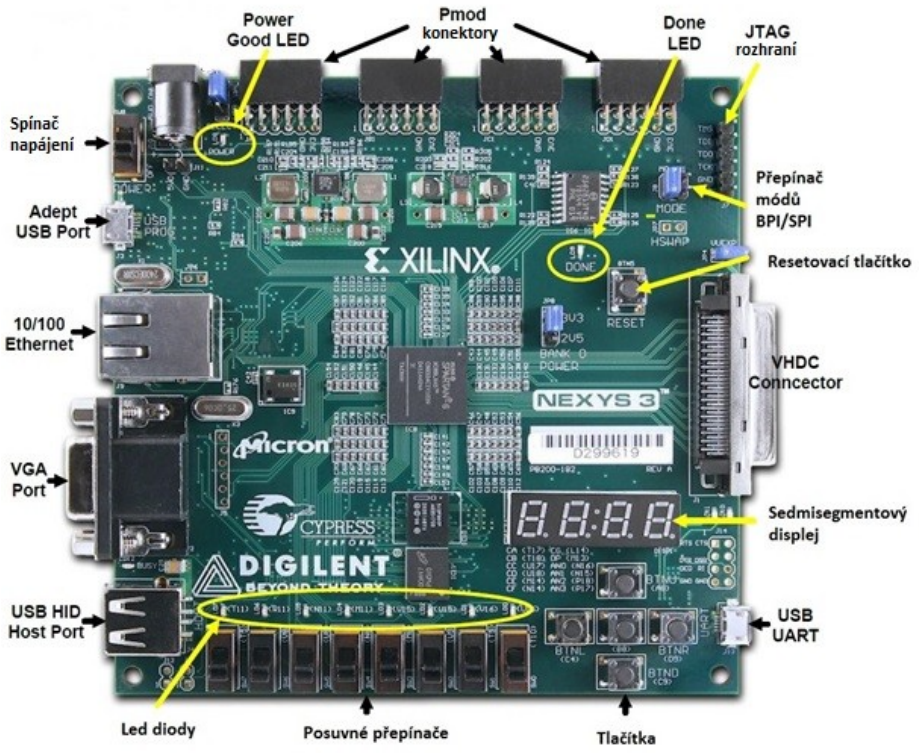
- LUT bloků (*Look Up Table*) – jsou to v podstatě 16bitové paměti schopné realizovat všechny logické funkce čtyř nebo více proměnných (záleží dle použité architektury)
- Klopných obvodů typu D (angl. FF – Flip-Flop) sloužících pro vytváření sekvenční logiky

- Paměť RAM – pro možnost ukládání dat
- Násobičky
- Generátor hodinových signálů [5]



Obrázek 2 – zobrazení vnitřní struktury FPGA

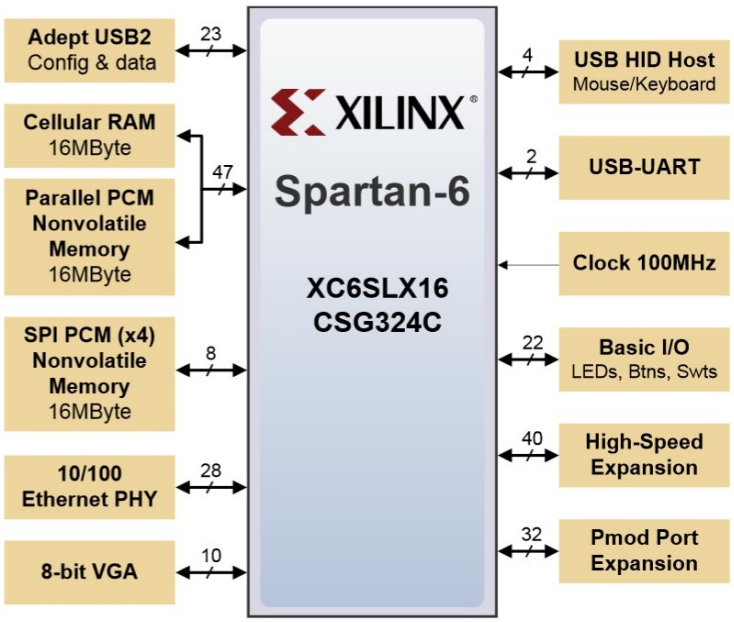
2.1.2 Vývojový kit NEXYS 3 a FPGA čip



Obrázek 3 – Vývojová deska NEXYS 3

Vývojový kit NEXYS 3, na kterém je realizována laboratorní úloha nabízí velmi bohatou podporu periferií:

- 100 MHz krystal – hodnota základního hodinového kmitočtu pro FPGA
- Paměť RAM – 16 MByte paralelní, nevolatilní paměť s rychlostí zápisu až 80MHz
- Adept USB port – programovací port USB 2.0 pro nahrání vytvořeného programu ve VHDL do FPGA
- 10/100 Ethernetový vstup pro komunikaci se sítí
- VGA port – 8bit VGA port pro připojení monitoru
- Tlačítka – naprogramováním se určí jejich funkce
- Switch posuvné spínače – funkce naprogramováním
- Signalizační led diody – funkce programovatelná, např: využití pro testování funkčnosti prvních programů
- Čtyřmístný sedmisegmentový displej
- FPGA čip XILINX Spartan-6 LX16 (obr. č. 4)



Obrázek 4 – Bloková struktura desky NEXYS 3 (převzato z [10])

Tabulka 1. – Přehled atributů zařízení Spartan-6 XC6SLX16 CSG324C [11]

FPGA čip		**Spartan-6 XC6SLX16 CSG324C**
Počet logických buněk[1]		14,579
Konfigurační logické bloky CLB	Logické buňky (Slices)[2]	2278
	D klopné obvody (FF Flip-Flops	18224
	RAM [Kb]	136
DSP48A1 slices – vysokorychlostní, nízkonapěťové logické buňky[3]		32
Počet paměťových RAM bloků	18 Kb	32
	Max (Kb)	576
Počet CMTs - *clock management tiles*[4] časových základen		2
Počet vstupně výstupních I/O částí		4
Maximální počet vstupů/výstupů		232

1. Spartan 6 FPGA nabízí 6vstupé LUT buňky.
2. Každá logická buňka (Slices) obsahuje 4 LUT buňky a 8 D-klopných obvodů
3. Každá DSP48A1 buňka obsahuje 18 x 18 násobičku, sčítačku a akumulátor.
4. Každá CMT obsahuje dva DCMs (Digital Clock Managers) bloky a jeden PLL (Phase-Locked Loops)

2.2 VHDL (Very High Speed Integrated Circuits – Hardware Description Language)

VHDL je jazyk zejména používaný pro popis hardwaru programovatelných logických obvodů a nepatří do skupiny standartních programovacích jazyků, musíme počítat s jistými omezeními, která jsou dána jeho těsnou vazbou na použitý typ programovatelného logického pole. [7]

Základní vlastnosti jazyka VHDL:

- Je to otevřený standard (*open standart*), tedy licence jeho vlastníka není třeba pro sestavení návrhových systémů, jako je tomu například u jazyka ABEL).

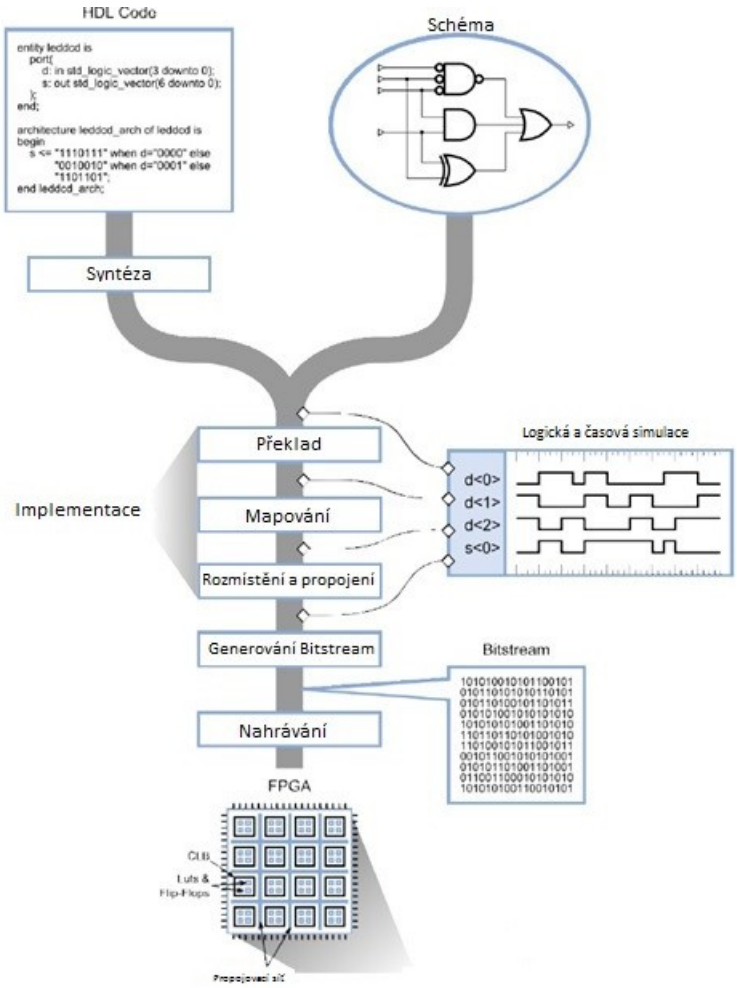
- I bez finálních požadavků na systém je možno na návrhu pracovat a ten přenášet z jednoho návrhového systému na jiný.

- Možnost provedení simulací zdrojového kódu v simulátorech a syntetizérech. [6]

2.2.1 Popis konstrukce VHDL ve vývojovém prostředí ISE Design Suite (v14.7)

Základní struktura modelu v jazyku VHDL má dvě základní části: **deklarace entity** a **tělo (popis) architektury.** Deklarací entity popisujeme vstupy a výstupy konstrukce a popisem architektury definujeme její funkci. [8]

Dále se otázkou jak postupovat při tvorbě vlastního VHDL kódu nebudeme zabývat, jelikož by to značně přesahovalo požadované limity téhle práce a také to není předmětem zadání.



Obrázek 5 – Zjednodušené schéma vývojového procesu v prostředí ISE Design Suite [12]

Po vytvoření vlastního VHDL kódu, přecházíme na proces implementace, kde dochází ke konverzi VHDL do konfiguračních dat pro programovatelné hradlové pole (Obr č. 5). Proces implementace je tvořen několika kroky:

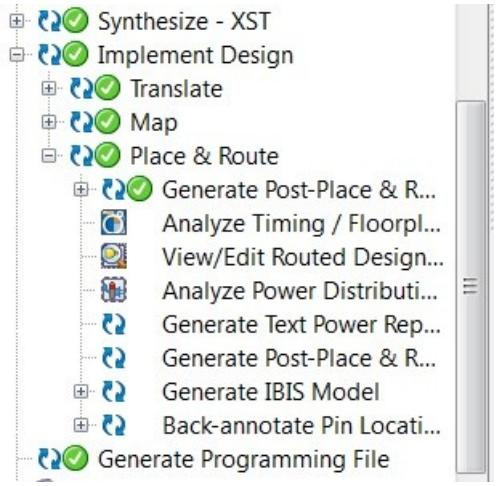
- Syntéza (synthesize) – jde o proces konverze zdrojového HDL kódu, do tímto procesem vytvořeného seznamu logických prvků a jejich vzájemného propojení (*netlist*). Dále je odstraněna nepoužitá logika a proces je optimalizován. Kromě HDL kódu je načtena také technologická knihovna.
- Překlad (*Translate*) – vstupem procesu je syntézou vytvořený *netlist* a soubor uživatelských omezení. Výstupem je tzv. NGD (*Native Generic Database*) file, kterým se dostáváme do další fáze.
- Mapování (mapping) – proces mapování načte procesem (*Translate*) vytvořený NGD soubor a zároveň logickou strukturu výsledného FPGA a vygeneruje NCD (*Native Circuit Description*) file – ten představuje fyzické rozložení prvků v obvodu, který je použit pro rozmístění a propojení a také PCF (*Physical constraints file*).

*Poznámka:*Při použití obvodu Spartan6, nebo vyšší, dojde už v procesu mapování k rozmístění součástek, což se týká našeho případu. Jinak až dále v procesu rozmístění a propojení.

- Proces rozmístění a propojení vykoná přesně tyhle úkony díky souboru NCD vygenerovaném v předchozím kroku.[12]

Celou bázi úkonů zakončíme kliknutím na Generate Programming File, čímž dostaneme požadovaný Bitstream, soubor s koncovkou **.bit**, který nahrajeme do desky NEXYS 3.

Ideálně bude na konci celý proces vypadat následovně:



Obrázek 6 – proces úspěšně vytvořeného Bitstreamu

3. Soubory omezení

Zavedení omezení je pokyn nástroji implementace k následování uživatelem a z části také programem zadaných omezení pro mapování, rozmístění, seskupení, či časování aj. [1]

Omezení se používají především k optimalizaci při rozmisťování, ale také k zajištění synchronních vstupů a řízení výstupu. Slouží ke specifikaci funkčních požadavků a definuje je uživatel. Poskytují cíle, které se optimalizace a mapování procesů snaží splnit. [1]

3.1 Prostředky pro zadávání omezení

Hlavním prostředkem pro zadávání omezení je tzv. UCF (*User Constraints File*) soubor.

- **UCF -** je v podstatě textovým dokumentem psaným v ASCII kódu a tvořen uživatelem. Načítán je procesem *Translate* a je zakomponován do NGD souboru (viz. odstavec Překlad).

Omezení můžeme také zadávat do souborů NCF (*Netlist Constraints File*) a PCF (Physical Constraints File).

- **NCF** – platí pro něj stejná pravidla jako v případě UCF s tím rozdílem že je vygenerován v procesu Syntéza
- **PCF** – je to soubor omezení vygenerován při procesu mapování a vzniká překladem pro již konkrétní architekturu ze souborů omezení *netlist* a UCF. Můžeme do něj tedy zapsat jakákoliv tzv. fyzická omezení (*Physical Constraints*). Nevýhoda zadání do PCF souboru je, že pokaždé když je proveden proces mapování je soubor přepsán.

Co však platí a je doporučováno, primárně by měl uživatel zadávat omezení do souboru UCF, z důvodu jeho upřednostnění programem ISE Design Suite a také podpory zadání téměř všech dostupných omezení.

Každé omezení se zadává pomocí **Syntaxe** – pravidlo pro zápis popisující tvar a sled jednotlivých znaků a jejich skupin. V téhle práci se budeme zabývat pouze VHDL syntaxí zadávanou do souboru UCF.

3.2 Typy uživatelských omezení

3.2.1 Nařízení pro proces Mapování (Mapping Directives)

Soubor omezení dává instrukce procesu mapování, který se je snaží splnit. Obsahuje následující omezení:

- Area Group
- BEL
- Block Name
- DCI Value
- Drive
- Fast
- Hierarchical Block Name
- Hierarchical Lookup Table Name
- HU Set
- IOB
- Input Output Block Delay
- Input Output Standard
- Keep
- Keeper
- Lookup Table Name
- Map
- No Delay
- Pulldown
- Pullup
- Relative Location
- Relative Location Origin
- Relative Location Range
- Save Net Flag
- Slew
- U Set
- Use Relative Location
- XBLKNM

3.2.2 Omezení rozmístění (*Placement Constraints*)

Soubor omezení popisuje umístění každého logického prvku v návrhu FPGA jako je:

- D klopné obvody (FFs)
- ROM a RAM paměti
- CLB buňky
- Vstupně/výstupní bloky

Obsaženy jsou následující omezení:

- Area Group (AREA_GROUP)
- BEL
- Location (LOC)
- Locate (LOCATE)
- Prohibit (PROHIBIT)

- Relative Location (RLOC)
- Relative Location Origin (RLOC_ORIGIN)
- Relative Location Range (RLOC_RANGE)
- Use Relative Location (USE_RLOC)

3.2.3 Nařízení pro proces Propojení (*Routing Directives*)

Omezení poskytují specifické cíle pro proces PAR (*Place and Route*) a obsahují následující omezení:

- Area Group (AREA_GROUP)
- Configuration Mode (CONFIG_MODE)
- Lock Pins (LOCK_PINS)

4. Časová omezení (Timing Constraints)

Jak již bylo popsáno výše struktura čipu FPGA je v dnešní době velmi bohatá. Obsahuje značné množství konfiguračních logických bloků, pamětí RAM, mnoho vstupů, výstupů, integrované časové základny a častokrát je čip zasazen do tzv. kitu, tedy vývojové desky, která možnosti použití dále rozšiřuje. Tato rozšíření mohou být tvořeny USB porty, Ethernetovými konektory, VGA porty atd.

Co je však nyní nutné si uvědomit, že všechny tyhle prvky spolu ve zdařilém funkčním návrhu komunikují pomocí vzájemných propojení.

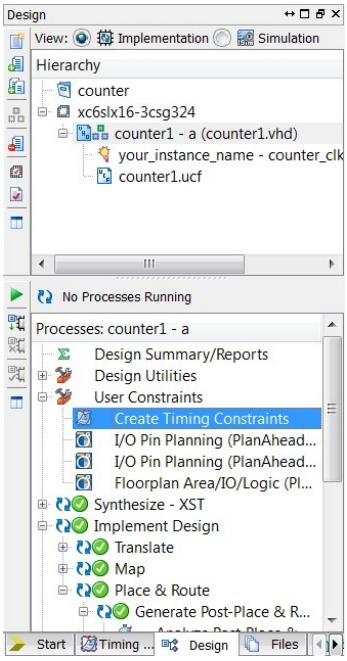
Existuje tu však velké riziko nekompatibility mezi hodinovými, či datovými signály přicházejícími na vstupy obvodu, také výstupy nemusí správně komunikovat s dalšími periferiemi a navíc vnitřní logika FPGA také může vykazovat jisté hazardní stavy při propojení složitých návrhů.

My ale všechno tohle, zda obvod bude, nebo nebude pracovat dle našich představ, můžeme ovlivnit. A to nejenom výslednou funkčnost, ale také určitou výkonnost v podobě velikosti hodnoty taktovacího signálu, tedy na jaké frekvenci bude obvod pracovat, dále množstvím použité logiky a v neposlední řadě můžeme předejít tvorbě nejistot, jenž také ovlivňují výslednou funkčnost návrhu.

Řešením k ošetření nežádoucích stavů je právě zakomponování časových omezení do návrhu a jednoznačně tak určit vztahy mezi jednotlivými prvky.

Software Xilinx umožňuje uživatelům zadat přesné časové požadavky pro jejich návrh. Tyto požadavky mohou být zadány pomocí globálních, anebo pro jednotlivé prvky v obvodu specifických omezení.

U každého omezení je uveden způsob zadání do UCF souboru pomocí specifické syntaxe, ale také pomocí softwarem ISE Design Suite 14.7 předpřipraveného grafického rozhranní s názvem „Create Timing Constraints“, jenž se nachází v okně Design>Processes>User Constraints. [2]



Obrázek 7 – položka Create Timing Constraints

4.1 Globální časová omezení

Můžeme říct, že uživatel si vystačí s globálními omezeními ve většině svých návrhů. Tato omezení tedy platí pro všechny cesty, na kterých jsou omezení možná.

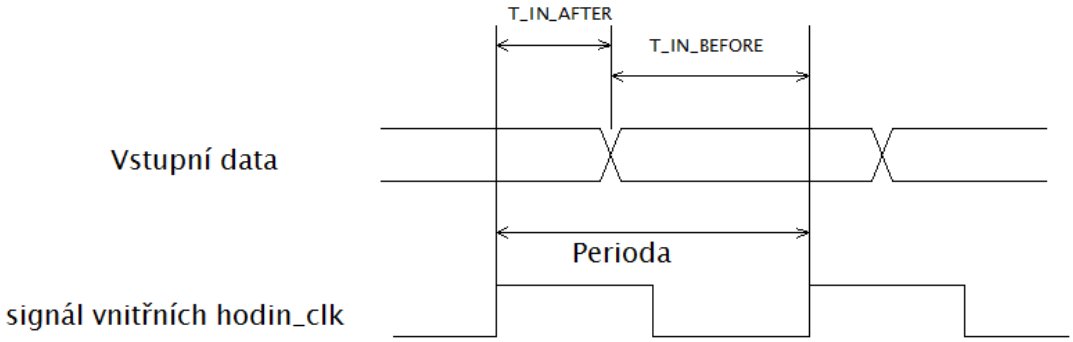
Díky globálním omezením jsme schopni ovlivnit časy:

- Vstupních cest (*Input paths*)
- Cest od synchronního prvku k synchronnímu prvku (*Register-to-register paths*)
- Výstupních cest (*Output paths*)
- Cest specifikovaných výjimkami (*Path specific exceptions*)

Tuhle kategorii představují následující omezení.

4.1.1 Vstupní časové omezení – OFFSET IN

Definuje vztah mezi vnitřními hodinami a daty přicházejícími s externího zařízení (nebo vstupního prvku) na vstupní piny FPGA. Jinými slovy se dá pomocí funkce OFFSET IN zajistit, aby data přicházející na vstup FPGA byla správně synchronizována s frekvencí vnitřních hodin, tím pádem správně načtena a zpracována.[2]



Obrázek 7 – schéma funkce OFFSET_IN

Obecná UCF Syntaxe:

OFFSET = IN "*offset_time* **"** [*units*] [**VALID** <*datavalid_time*> [*UNITS*]] {**BEFORE**|**AFTER**}

"*clk_name***"** [{**RISING**|**FALLING**}]**;**

- offset_time [units] – rozdíl mezi hranou hodin a počátkem zachycením dat, tedy o kolik má být data nachystána, zpravidla [ns, ps]
- datavalid_time [units] – zde zapisujeme číslo, po jakou dobu jsou data platná, opět zpravidla [ns, ps]
- BEFORE/AFTER – výběrem jedné s možností určujeme, zda mají být data nachystána před nebo po hraně hodin
- clk_name – uvedeme název vnitřní hodinové sítě
- RISING/FALING – výběrem jedné s možností určíme, zda má funkce reagovat na vzestupnou, nebo sestupnou hranu hodinového signálu [2]

Příklad:

NET "clock" TNM<_NET = CLK;

TIMESPEC TS_CLK = PERIOD CLK 5.0 ns HIGH 50%;

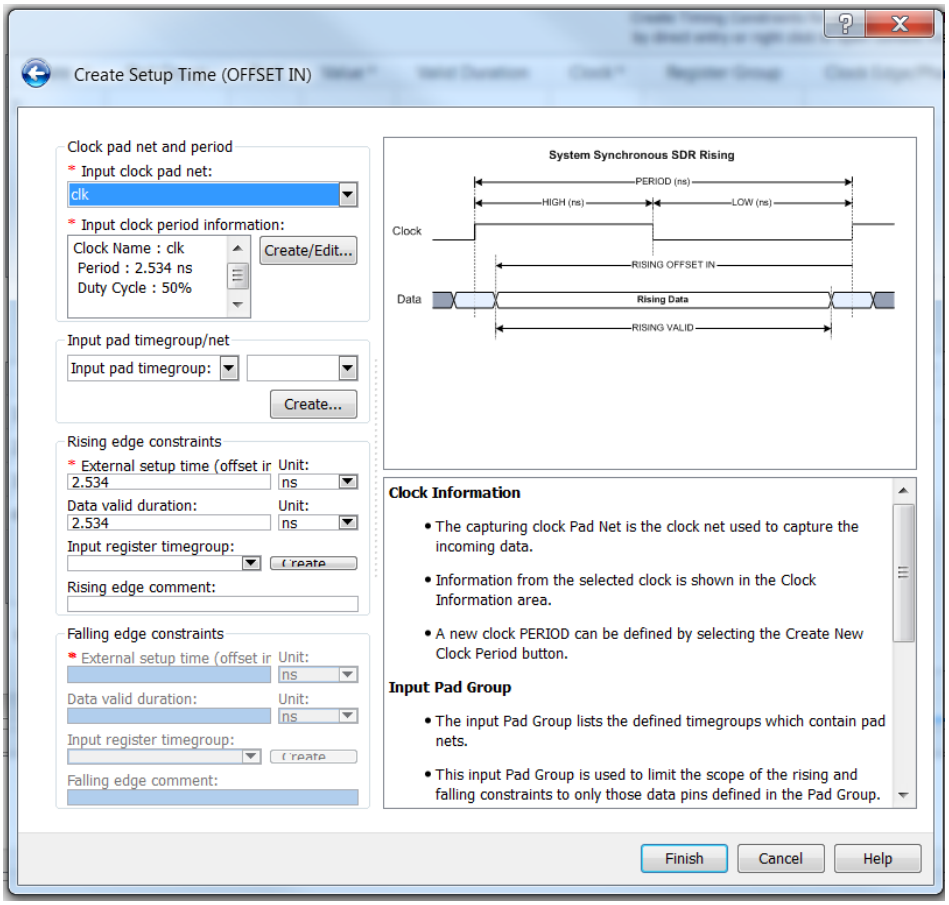
OFFSET = IN 1 ns VALID 3 ns BEFORE clock RISING;

Vysvětlení:

Pro lepší demonstraci je uvedena také perioda 5 ns s logickou "1" po dobu 2,5 ns a název hodinové sítě tedy clock. Pro přicházející data tedy platí, že jsou nachystána 1 ns před vzestupnou hranou hodin po dobu platnosti 3 ns.

Nastavení OFFSET IN pomocí rozhranní Create Timing Constraints v softwaru ISE Design Suite 14.7:

Začneme rozbalením záložky Timing Constraints a dvojklikem na Inputs a zobrazí se nám okno s možnostmi nastavení parametrů OFFSET IN.

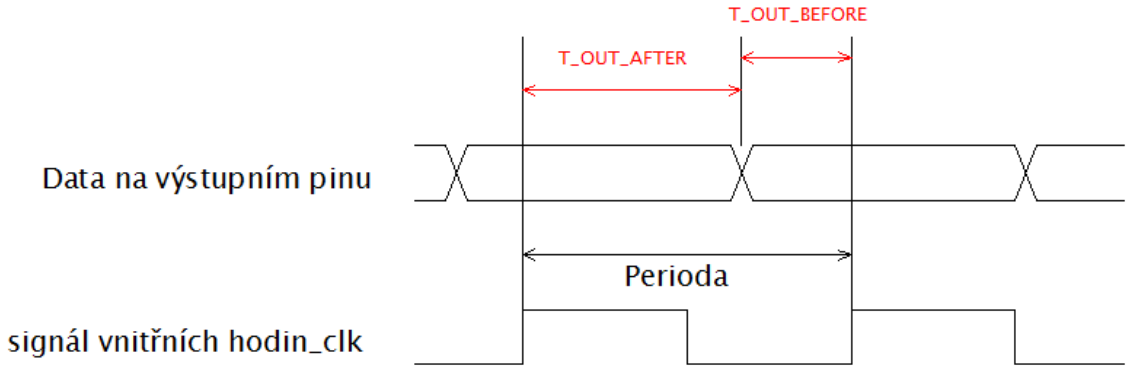


Obrázek 8 – okno pro nastavení parametrů omezení OFFSET IN

Samotná změna nastavení hodnoty omezení OFFSET IN se provádí v kolonce External setup time offset in. Okno nám nabízí daleko více možností nastavení, než jenom OFFSET IN. Můžeme určit dobu platnosti dat, ale také vytvořit skupinu z prvků D-klopných obvodů, nebo už určit existující, pro kterou má zadané omezení OFFSET IN platit a to výběrem s rolovací lišty pod Input register timegroup, nebo kliknutím na create.

4.1.2 Výtupní časové omezení – OFFSET OUT

Definuje vztah mezi vnitřními hodinami a daty na výstupním pinu, která mohou být dále využita, buď vnitřní strukturou, nebo externím zařízením.



Obrázek 9 – schéma funkce OFFSET_OUT

Obecná UCF Syntaxe:

OFFSET = OUT "*offset_time***" [***units***] {BEFORE "***clk_name***"| AFTER "***clk_name***"} [{RISING | FALLING}];**

- offset_time [units] – rozdíl mezi hranou hodin a časem, kdy se mají data objevit na výstupním pinu. Jednotky zpravidla [ns, ps].
- BEFORE/AFTER - výběrem jedné s možností určujeme, zda mají být data nachystána před nebo po hraně hodin
- clk_name – uvedeme název vnitřní hodinové sítě
- výběrem jedné s možností určíme, zda má funkce reagovat na vzestupnou, nebo sestupnou hranu hodinového signálu [2]

Příklad:

NET "clock" TNM_NET = CLK;

TIMESPEC TS_CLK = PERIOD CLK 5.0 ns HIGH 50%;

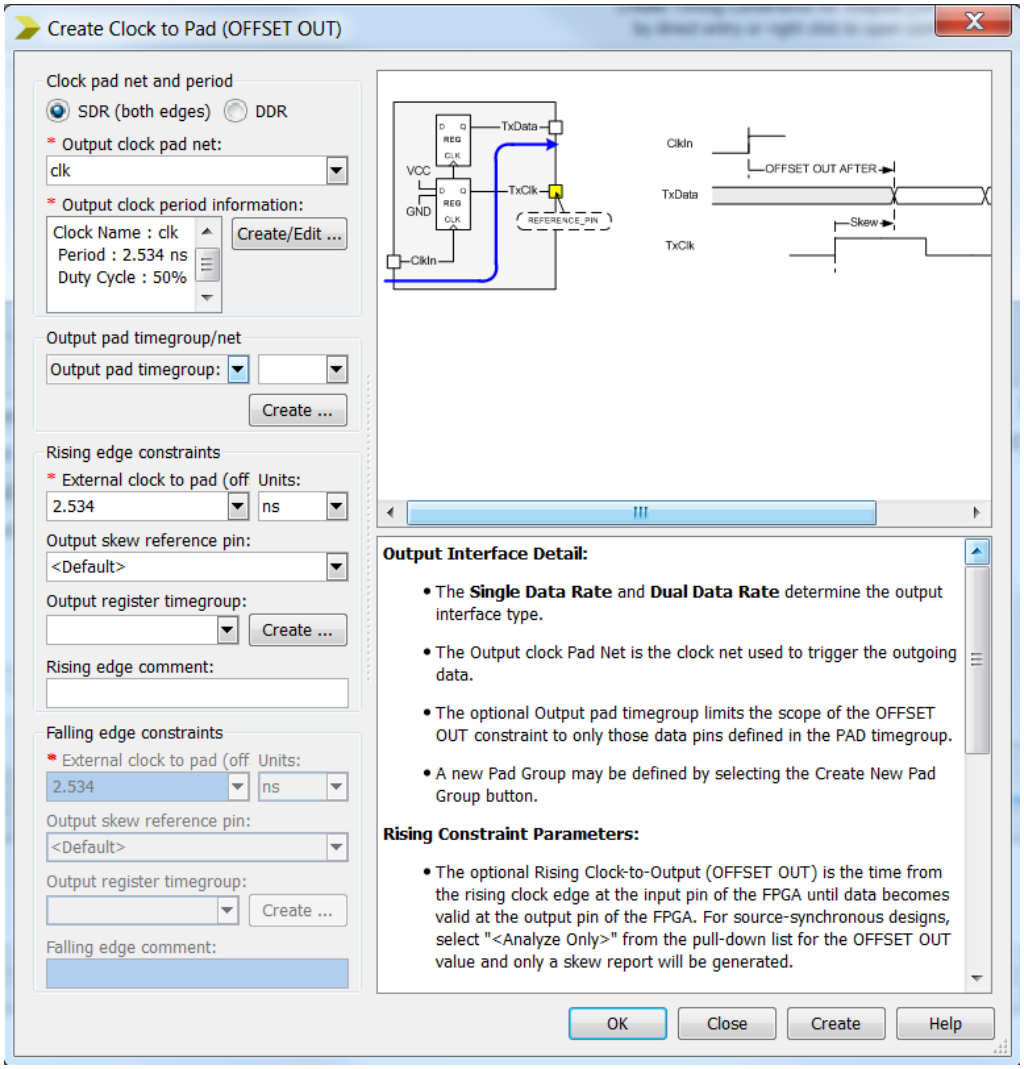
OFFSET = OUT 5 ns AFTER clock FALLING;

Vysvětlení:

Pro lepší demonstraci je uvedena také perioda 5 ns s logickou "1" po dobu 2,5 ns a název hodinové sítě tedy clock. Pro data tedy platí, že jsou na výstupním pinu 1 ns po sestupné hraně hodin.

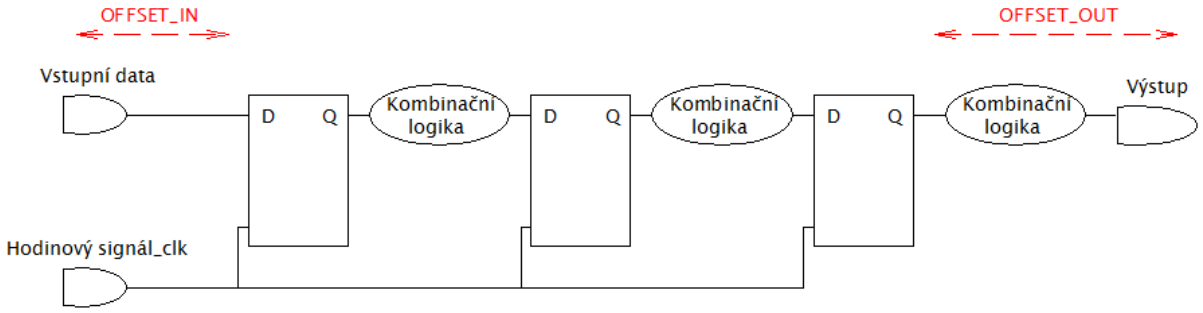
Nastavení OFFSET OUT pomocí rozhranní Create Timing Constraints v softwaru ISE Design Suite 14.7:

Začneme rozbalením záložky Timing Constraints a dvojklikem na Outputs, zobrazí se nám okno s možnostmi nastavení parametrů OFFSET OUT.



Obrázek 10 – okno pro nastavení parametrů omezení OFFSET OUT

Změnit hodnotu nastavení omezení OFFSET OUT můžeme v kolonce External clock to pad offset out a třeba vytvořit, nebo vybrat skupinu z D-klopných obvodů pro kterou by platila specifikovaná omezení OFFSET OUT můžeme v kolonce Output register timegroup, nebo kliknutím na tlačítko create.



Obrázek 11 – zjednodušené schéma použití funkce OFFSET

4.1.3 Omezení mezi synchronními prvky – PERIOD

Tohle omezení, jak už z názvu vyplývá, nám umožňuje upravit délku periody, tedy čas, s jakým se hodinový signál periodicky překlopí z logické úrovně ”1” do logické úrovně ”0” zpět. Zahrnuje všechny synchronní prvky připojené k hodinovému signálu.

Tohle je to TOP, to nejzákladnější omezení, od kterého se odvíjí, nebo na něj navazují všechny další prvky a v podstatě nám udává hodnotu frekvence, na jaké může obvod pracovat.

Vztah frekvence a periody:

$$f = \frac{1}{T} \tag{1}$$

Obecná UCF Syntaxe:

TIMESPEC “TS*identifier***”=PERIOD “***TNM_reference***”** *period* [*units*] {**HIGH** | **LOW**}

[*high_or_low_time*] **INPUT_JITTER value;**

- identifier – pro zadání unikátního názvu sítě
- TNM_reference – tímhle názvem můžeme identifikovat skupinu, pro kterou bude perioda platit, typicky TNM_NET.
- period – zadaná hodnota periody, bez zadání jednotek, jsou automaticky nastaveny na ns, jinak můžeme zadat ps, ns, micro, ms. Jednotka periody může být specifikována také pomocí hodnoty frekvence jako kHz, MHz, GHz.
- HIGH/LOW – určuje hodnotu prvního pulzu periody. Pro HIGH je logická ”1”, pro LOW logická ”0”.
- high_or_low_time – zadáváme hodnotu času, po který je nastavena buď úroveň HIGH, nebo LOW, běžně se také udává v procentech, jako tzv. střída.
- INPUT_JITTER – zadáváme dobu, za kterou trvá změna signálu z logické ”0” do logické ”1”, nebo opačně. Bez zadání jednotky automaticky nastavena na ps.[2]

Příklad:

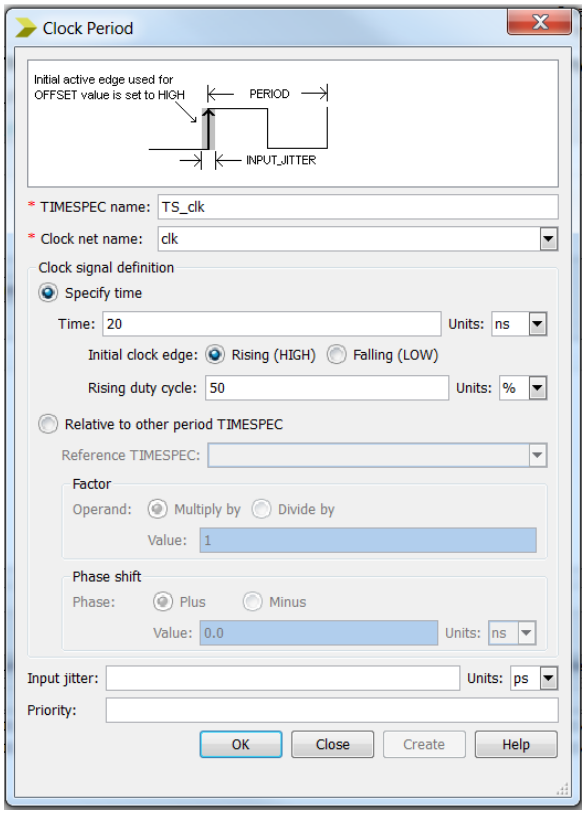
TIMESPEC TS_clk = PERIOD “clk” 20 ns HIGH 50% INPUT_JITTER 500 ps;

Vysvětlení:

Hodinový signál clk, s periodou 20 ns, střídou 50% a dbou změny log. úrovně 500 ps.

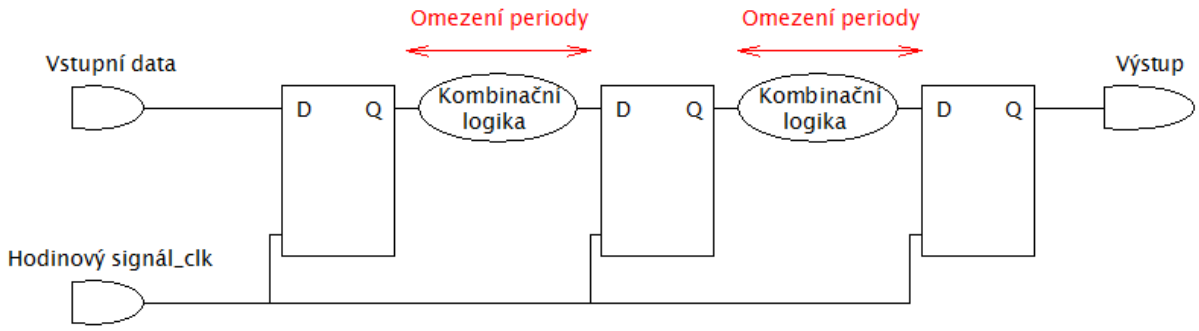
Nastavení periody pomocí rozhranní Create Timing Constraints v softwaru ISE Design Suite 14.7:

Začneme rozbalením záložky Timing Constraints a kliknutím na Clock Domains, pokračujeme dvojklikem na hodinový signál, u kterého chceme hodnotu periody nastavit.



Obrázek 12 – formulář pro nastavení periody

Při označeném výběru Specify time můžeme nastavit hodnotu periody v kolonce Time, následuje výběr hrany a střída. Nahoře v kolonce Clock net name si můžeme ještě vybrat, pro jaký hodinový signál periodu nastavujeme.



Obrázek 13 – zjednodušené schéma použití funkce PERIOD

4.1.4 Omezení specifikací cesty – FROM:TO

Jedná se o omezení cest, na kterých je povoleno více hodinových signálů, tyhle typy cest jsou běžně zahrnuty do omezení pomocí periody, avšak jejich zahrnutím do specifického omezení jsou z vlivu omezení periodou vyjmuty, aby nedocházelo k chybám.

Mohou být tedy taktovány stejným hodinovým signálem, ale pracovat o jiné periodě.

Omezení je možno zadávat mezi programem připravenými skupinami prvků jako (vstupně/výstupní piny, D-klopné obvody, paměti RAM aj.), nebo skupinami uživatelem vytvořenými. To vše nezávisle na hodinovém signálu.

Obecná UCF Syntaxe:

TIMESPEC TS*name***=FROM "***group1***" TO "***group2***"** *value* [**DATAPATHONLY**]**;**

- **TS**name – určení názvu omezení, vždy musí začínat na TS.
- group1 – názvem skupiny 1 se určí počátek cesty
- group2 – názvem skupiny 2 se určí cíl cesty
- value – určení časové jednotky, jako výchozí jsou nastaveny ns
- DATAPATHONLY – tohle klíčové slovo znamená, že omezení FROM:TO nebere v potaz časovou délku změny hrany (clock skew) ani fázi (clock phase). [2]

Příklad:

TIMESPEC TS_P2F = FROM PADS TO FFS 8 ns;

TIMESPEC TS_F2F = FROM FFS TO FFS 5 ns;

TIMESPEC TS_F2P = FROM FFS TO PADS 14 ns;

TIMESPEC TS_P2P = FROM PADS TO PADS 12 ns;

Vysvětlení:

Omezení P2F mezi vstupním pinem a D klopným obvodem nastaveno na 8 ns.

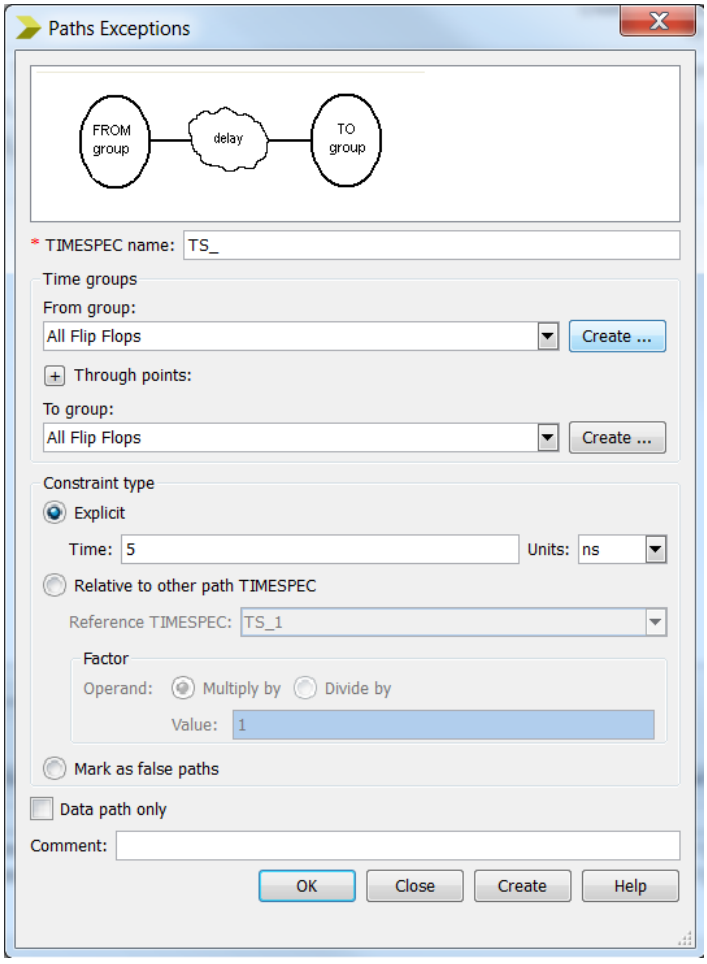
Omezení F2F mezi dvěma D klopnými obvody nastaveno na 5 ns.

Omezení F2P mezi D klopným obvodem a výstupním pinem nastaveno na 14 ns.

Omezení P2P mezi vstupně/výstupními piny nastaveno na 8 ns.

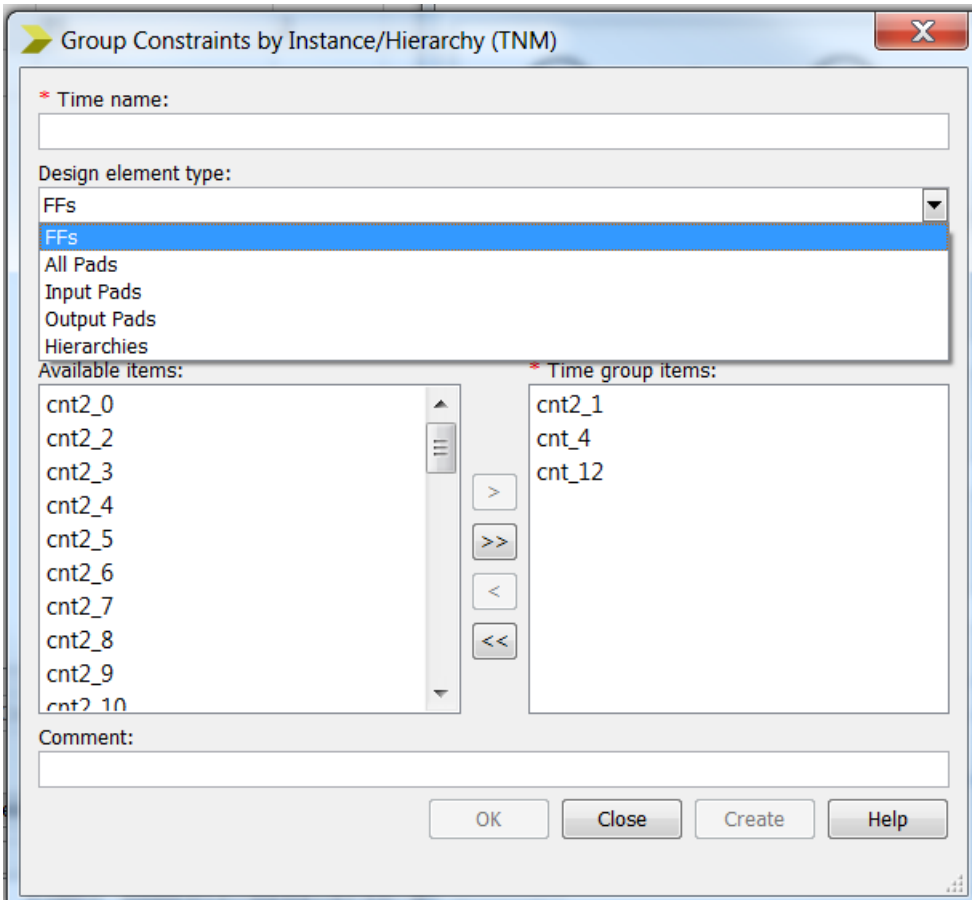
Nastavení hodnoty omezení FORM:TO pomocí rozhranní Create Timing Constraints v softwaru ISE Design Suite 14.7:

Začneme rozbalením záložky Timing Constraints, rozbalením Exceptions a pokračujeme dvojklikem na Paths, kde nastavíme hodnotu omezení, ale také mezi kterými z dostupných skupin má být omezení platné.

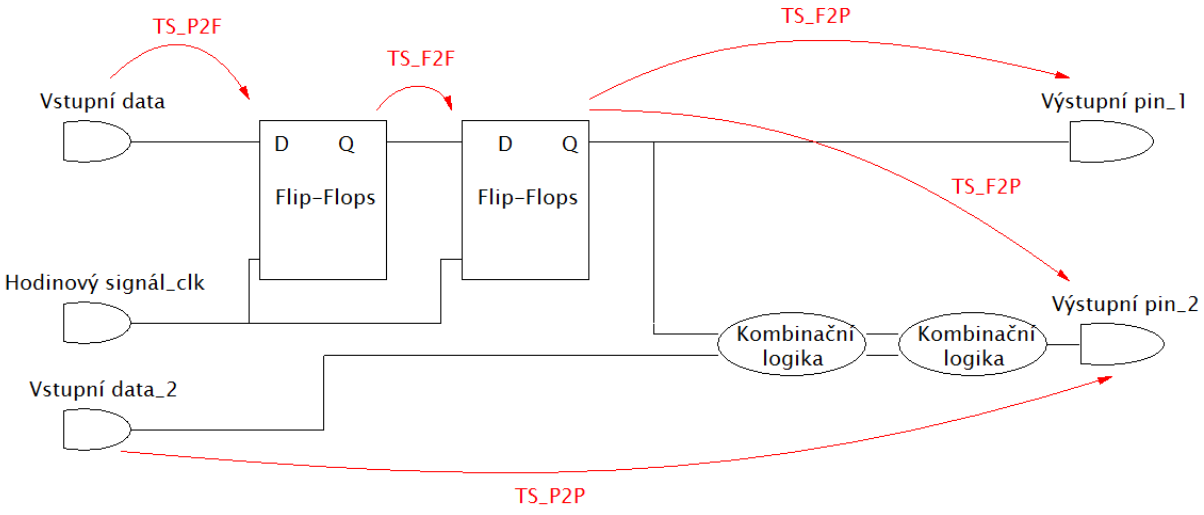


Obrázek 14 – nastavení omezení F2F na hodnotu 5 ns

Kliknutím na tlačítko Create u výběru skupiny se otevře nové okno s možností vytvořit si vlatní skupinu prvků u kterých chceme omezení FROM:TO nastavit.



Obrázek 15 Tvorba vlastní skupiny vybraných prvků



Obrázek 16 – zjednodušené schéma použití funkce FROM:TO

Omezení může být ještě zadáno následujícími kombinacemi:

- From:To
- From:Thru:To
- Thru:To
- From:Thru
- From
- To
- From:Thru:Thru:Thru:To

V podstatě jde o upravení funkce přidáním průchozích bodů THRU.

4.2 Další časová omezení

Pokud ani po zadání globálních omezení platící pro všechny prvky obvodu (s výjimkami) není uživatel s výsledným návrhem spokojen. Je možno přikročit k zadání dalších dostupných omezení.

4.2.1 MAXDELAY (Maximum Delay)

Definuje maximální zpoždění vybraného prvku na propojovací síti.

Obecná UCF Syntaxe:

NET "*net_name***" MAXDELAY=***value units***;**

- net_name – název prvku
- value – číselná kladná hodnota zpoždění a doplnění časové jednotky, výchozí nastavená je ns

Příklad:

NET "cnt<0>" **MAXDELAY =** 2 ns;

4.2.2 TIG (Timing Ignore)

Zahrnutím cest do omezení Timing Ignore způsobí, že s nimi implementační nástroj bude zacházet, jako kdyby omezení neexistovala. Je používáno především v případech, kdy se nám vyskytne chyba v návrhu. Chybou můžeme myslet třeba nesplnění časových podmínek jednoho nebo více prvků

v obvodu. Zahrnutím prvku, nebo cesty do omezení Timing Ignore jej pomyslně vyjmeme z časové analýzy při řešení vyskytlých chyb na spojích, cestách a pinech obvodu. [2]

UCF Syntaxe:

NET " *net_name***" TIG = TS**identifier**;**

- místo NET můžeme také napsat PIN, PATH, TIMEGRP aj.
- net_name – název prvku
- **TS**identifier – časová specifikace, která bude ignorovaná.

4.2.3 Priority

Pokročilé omezení PRIORITY je používáno pro případ, že jsou pro stejnou cestu užita dvě časová omezení.

Příklad UCF Syntaxe:

• TIMESPEC TS_01 = FROM A_grp TO B_grp 10 ns PRIORITY 5;

• TIMESPEC TS_02 = FROM A_grp TO B_grp 20 ns PRIORITY 1;

Pro skupinu A a B bude platit omezení TS_02, jelikož má vyšší prioritu. Čím nižší číslo v rozsahu 255 až -255, tím vyšší priorita.

4.2.4 Skupinová omezení (*Grouping Constraints*)

Jako další musí být do kategorie časových omezení zahrnuta také skupinová omezení, jež se velkou měrou podílejí na možnostech tvorby výsledného časového návrhu.

Představiteli skupinových omezení jsou:

- Timing Name (TMN)
- Timing Name Net (TMN_NET)
- Timing Group (TIMEGRP)

Všechny se používají obdobně, zkrátka se pomocí nich dají vytvářet skupiny pinů, sítí a jednotlivých prvků, pro které je možno souhrnně vytvářet časová omezení.

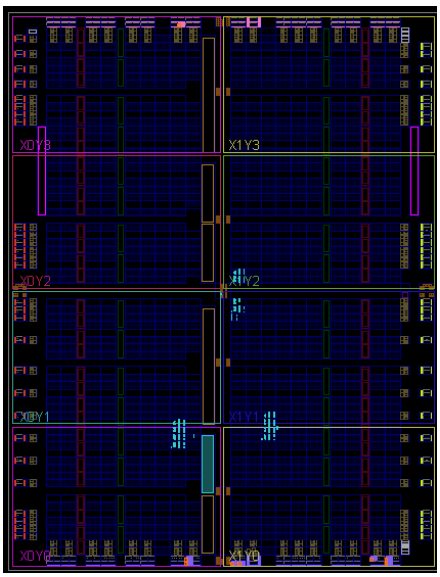
U veškerých příkladů jsou uvedené časy pouze vysvětlující, pro opravdové zjištění časů zpoždění v návrhu se používá, v okně Design Summary nástroje Static Timing a jeho bohatého reportu.[2]

5. Návrh koncepce úlohy pro demonstraci vlivu časových omezení

Jedno z hlavních kriterií, jenž by měla úloha demonstrovat byla myšlenka posunout návrh díky časovým omezením vpřed, zkrátka zrychlit ho, zefektivnit. Z toho důvodu je v laboratorní úloze také zakomponován prvek DCM (Digital Clock Manager) generátor hodinových pulsů, díky němuž, jednak nám návrh nabývá na objemu použité logiky a také je to součástka s některými pevně danými časovými zpožděními vzhledem třeba k základnímu hodinovému signálu. Už jen zakomponováním DCM do obvodu nám vznikl problém, kterým se zabývá určitý bod zadání laboratorní úlohy a vede osoby, které ji budou měřit také k zjištění, že ne všechno se dá časově omezit.

Funkce úlohy spočívá ve třech čtrnácti bitových čítačích, avšak s deklarovaným typem INTEGER, což nám poskytuje dostatek logiky pro možné představení a vlivu časových omezení. Jejich funkci na vývojové desce můžeme vidět díky k nim připojeným led diodám, vlastně jakmile dočítají do konečného stavu s tím, že jim zároveň budou povoleny hodiny, uvidíme signalizační led, že svého konečného stavu opravdu dosáhly.

Velkou měrou se na návrhu úlohy pro časová omezení podílí také funkce zjištění časových prametrů, která je představena nástrojem Static Timing. Ten udává opravdu obsáhlé informace o časování na tímto časováním omezených spojích. Také na orientaci v Static Timing reportu je zaměřen jeden z bodů laboratorní úlohy. Výpis Static Timing reportu je uveden v příloze, jedná se o počáteční stav laboratorní úlohy.



Obrázek 17 Zobrazení rozložení logiky v PlanAhead

Dalším neodmyslitelným bodem koncepce úlohy musí být nástroj PlanAhead, díky kterému můžeme vidět využití jednotlivých CLB buňek, vstupně výstupních pinů či zmiňovaných DCM bloků a jiné

použité logiky. Zkrátka si velmi dobře udělat představu o tom, jak se při určitém návrhu na FPGA čipu logika zapojila (viz obrázek 17).

Tabulka 2 – Přehled využití logiky v návrhu laboratorní úlohy před omezením

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	45	18,224	1%	
Number used as Flip Flops	45			
Number of Slice LUTs	99	9,112	1%	
Number used as logic	96	9,112	1%	
Number using O6 output only	54			
Number using O5 output only	39			
Number using O5 and O6	3			
Number used exclusively as route-thrus	3			
Number with same-slice carry load	3			
Number of occupied Slices	27	2,278	1%	
Number of MUXCYs used	48	4,556	1%	
Number of LUT Flip Flop pairs used	99			
Number with an unused Flip Flop	54	99	54%	
Number with an unused LUT	0	99	0%	
Number of fully used LUT-FF pairs	45	99	45%	
Number of unique control sets	3			
Number of slice register sites lost to control set restrictions	3	18,224	1%	
Number of bonded IOBs	8	232	3%	
Number of LOCed IOBs	8	8	100%	
Number of BUFIO2/BUFIO2_2CLKs	1	32	3%	
Number used as BUFIO2s	1			
Number of BUFIO2FB/BUFIO2FB_2CLKs	1	32	3%	
Number used as BUFIO2FBs	1			
Number of BUFG/BUFGMUXs	1	16	6%	
Number used as BUFGs	1			
Number of DCM/DCM_CLKGENs	1	4	25%	
Number used as DCMs	1			
Average Fanout of Non-Clock Nets	4.08			

6. Závěr a zhodnocení dosažených výsledků

Po seznámení se s technikou FPGA a návrhem programovatelných logických obvodů, jsem nastudoval a zde uvedl problematiku omezení při návrhu FPGA. Zdroji informací mi byly především dokumenty „User Guide“ firmy XILINX Inc. ze kterých jsem mohl čerpat množství komplexních i velmi specifických informací nejenom ohledně časových omezení, ale také všeobecně k tématu FPGA a programovacího jazyka VHDL.

Hlavním cílem mé práce bylo vytvoření laboratorní úlohy, na které se studenti mohou blíže seznámit s časovými omezeními a také si vyzkoušet jejich vlivy na výsledný návrh, které spočívají převážně v optimalizaci rozmístění použité vnitřní struktury v čipu FPGA a tím vedou k rapidnímu zvýšení frekvence, na které je obvod schopen pracovat. Při vypracování laboratorní došlo ke zvýšení o 230 MHz při použití několika nejdůležitějších časových omezení. Další požadovaná omezení by byla přes rámec výukové hodiny, jelikož téma a možnosti omezení jsou příliš rozsáhlá. Vypracovaná laboratorní úloha je uvedena v příloze na CD.

Práce může být využita jako český návod pro pochopení a uvedení do problematiky časových omezení, jelikož v současné době se žádná práce v češtině s tématem časových omezení nevyskytuje. Do budoucna tak může sloužit jako základ mě, nebo dalším studentům k rozsáhlejšímu a hlubšímu probádání podobného tématu.

7. Použitá literatura

[1]. XILINX Inc. *Constraints Guide: UG625*. 1.4. 2013. Dostupné z: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/cgd.pdf

[2]. XILINX Inc. *Timing Closure User Guide: UG612*. 16.10.2012. Dostupné z: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/ug612.pdf

[3]. PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL*. 1. vyd. Praha: BEN - technická literatura, 2006, 349 s. ISBN 80-730-0198-5.

[4]. KAŠÍK V. *Programovatelná hradlová pole FPGA* [učební text a návody do cvičení]. Ostrava: VŠB – TU, Fakulta elektrotechniky a informatiky, 2013. 62 s.

[5]. ŠŤASTNÝ, Jakub. *FPGA prakticky: realizace číslicových systémů pro programovatelná hradlová pole*. 1. vyd. Praha: BEN - technická literatura, 2010, 199 s. ISBN 978-80-7300-261-9.

[6]. Musil. V, Kolouch. J, Prokop. R. *Návrh digitálních integrovaných obvodů a jazyk VHDL*.[skriptum], Vyd. Brno, 2002,185 s.

[7]. Lafata P. *Pokročilé využití jazyka VHDL*. [skriptum]. Praha: ČVUT, Fakulta elektrotechnická, 33 s.

[8]. Kolouch J. *Jazyk VHDL a jeho užití pro syntézu číslicových systémů*.[skriptum]. Brno 2005

[9]. PERRY, Douglas L. *VHDL: programming by example*. 4th ed. New York: McGraw-Hill, c2002, xvii, 476 s. ISBN 00-714-0070-2.

[10]. DIGILENT. *Nexys 3 reference manual*. 10. 4. 2013. Dostupné z: http://digilentinc.com/Data/Products/NEXYS3/Nexys3_rm_V2.pdf

[11]. XILINX INC. *Spartan-6 Family Overview: DataSheet160 (v2.0)*. 25. 10. 2011. Dostupné z: http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[12]. XILINX INC. *Command Line Tools User Guide: UG628 (v14.7)*. 2. 10. 2013. Dostupné z: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/devref.pdf

8. Přílohy

I.	Laboratorní úloha_užití časových omezení v FPGA	Počet stran 10
II.	Vypracovaná laboratorní úloha_užití časových omezení v FPGA	Příloha na CD
III.	Vytvořený program v software ISE Design Suite 14.7	Příloha na CD
IV.	Výčet veškerých dostupných omezení	Příloha na CD

Obsah CD

BP_Jan_Nozicka.pdf

BP_Jan_Nozicka_Prilohy.zip

I. Laboratorní úloha – Užití časových omezení v FPGA

1. Teoretický úvod

Jak již bylo popsáno výše struktura čipu FPGA je v dnešní době velmi bohatá. Obsahuje značné množství konfiguračních logických bloků, pamětí RAM, mnoho vstupů, výstupů, integrované časové základny a častokrát je čip zasazen do tzv. kitu, tedy vývojové desky, která možnosti použití dále rozšiřuje. Tato rozšíření mohou být tvořeny USB porty, Ethernetovými konektory, VGA porty atd.

Co je však nyní nutné si uvědomit, že všechny tyhle prvky spolu ve zdařilém funkčním návrhu komunikují pomocí vzájemných propojení.

Existuje tu však velké riziko nekompatibility mezi hodinovými, či datovými signály přicházejícími na vstupy obvodu, také výstupy nemusí správně komunikovat s dalšími periferiemi a navíc vnitřní logika FPGA také může vykazovat jisté hazardní stavy při propojení složitých návrhů.

My ale všechno tohle, zda obvod bude, nebo nebude pracovat dle našich představ, můžeme ovlivnit. A to nejenom výslednou funkčnost, ale také určitou výkonnost v podobě velikosti hodnoty taktovacího signálu, tedy na jaké frekvenci bude obvod pracovat, dále množstvím použité logiky a v neposlední řadě můžeme předejít tvorbě nejistot, jenž také ovlivňují výslednou funkčnost návrhu.

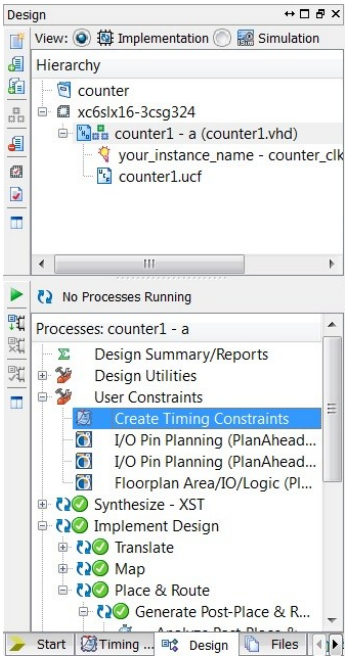
Řešením k ošetření nežádoucích stavů je právě zakomponování časových omezení do návrhu a jednoznačně tak určit vztahy mezi jednotlivými prvky.

Software Xilinx umožňuje uživatelům zadat přesné časové požadavky pro jejich návrh. Tyto požadavky mohou být zadány pomocí globálních, anebo pro jednotlivé prvky v obvodu specifických omezení.

2. Možnosti omezení a nastavení parametrů hodinového signálu.

Omezení, která budete potřebovat pro zvládnutí laboratorní úlohy:

U každého omezení je uveden způsob zadání do UCF souboru pomocí specifické syntaxe, ale také pomocí softwarem ISE Design Suite 14.7 předpřipraveného grafického rozhranní s názvem „Create Timing Constraints“,jenž se nachází v okně Design>Processes>User Constraints.



Obrázek 1 – položka Create Timing Constraints

2.1 Omezení mezi synchronními prvky – PERIOD

Tohle omezení, jak už z názvu vyplývá, nám umožňuje upravit délku periody, tedy čas, s jakým se hodinový signál periodicky překlopí z logické úrovně ”1” do logické úrovně ”0” zpět. Zahrnuje všechny synchronní prvky připojené k hodinovému signálu.

Obecná UCF Syntaxe:

TIMESPEC “TS*identifier***”=PERIOD “***TNM_reference***”** *period* [*units*] {**HIGH** | **LOW**}

[*high_or_low_time*] **INPUT_JITTER value;**

- identifier – pro zadání unikátního názvu sítě
- TNM_reference – tímhle názvem můžeme identifikovat skupinu, pro kterou bude perioda platit, typicky TNM_NET.
- period – zadaná hodnota periody, bez zadání jednotek, jsou automaticky nastaveny na ns, jinak můžeme zadat ps, ns, micro, ms. Jednotka periody může být specifikována také pomocí hodnoty frekvence jako kHz, MHz, GHz.
- HIGH/LOW – určuje hodnotu prvního pulzu periody. Pro HIGH je logická ”1”, pro LOW logická ”0”.
- high_or_low_time – zadáváme hodnotu času, po který je nastavena buď úroveň HIGH, nebo LOW, běžně se také udává v procentech, jako tzv. střída.

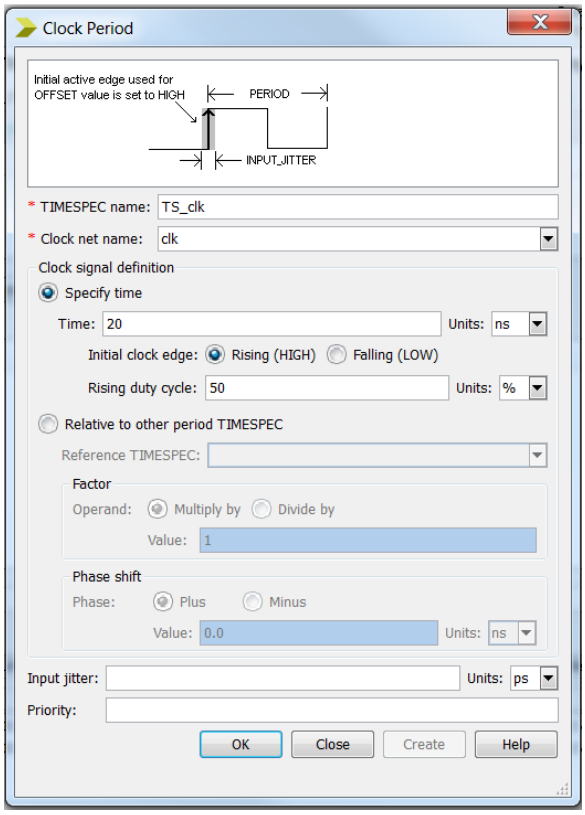
- INPUT_JITTER – zadáváme dobu, za kterou trvá změna signálu z logické ”0” do logické ”1”, nebo opačně. Bez zadání jednotky automaticky nastavena na ps.

Příklad:

TIMESPEC TS_clk = PERIOD “clk” 20 ns HIGH 50% INPUT_JITTER 500 ps;

Vysvětlení:

Hodinový signál clk, s periodou 20 ns, střídou 50% a dbou změny log. úrovně 500 ps.



Obrázek 2 Rozhraní pro nastavení periody

2.2 Omezení specifikací cesty – FROM:TO

Jedná se o omezení cest, na kterých je povoleno více hodinových signálů, tyhle typy cest jsou běžně zahrnuty do omezení pomocí periody, avšak jejich zahrnutím do specifického omezení jsou z vlivu omezení periodou vyjmuty, aby nedocházelo k chybám.

Mohou být tedy taktovány stejným hodinovým signálem, ale pracovat o jiné periodě.

Omezení je možno zadávat mezi programem připravenými skupinami prvků jako (vstupně/výstupní piny, D-klopné obvody, paměti RAM aj.), nebo skupinami uživatelem vytvořenými. To vše nezávisle na hodinovém signálu.

Obecná UCF Syntaxe:

TIMESPEC TS*name***=FROM "***group1***" TO "***group2***"** *value* [**DATAPATHONLY**]**;**

- **TS**name – určení názvu omezení, vždy musí začínat na TS.
- group1 – názvem skupiny 1 se určí počátek cesty
- group2 – názvem skupiny 2 se určí cíl cesty
- value – určení časové jednotky, jako výchozí jsou nastaveny ns
- DATAPATHONLY – tohle klíčové slovo znamená, že omezení FROM:TO nebere v potaz časovou délku změny hrany (clock skew) ani fázi (clock phase).

Příklad:

TIMESPEC TS_P2F = FROM PADS TO FFS 8 ns;

TIMESPEC TS_F2F = FROM FFS TO FFS 5 ns;

TIMESPEC TS_F2P = FROM FFS TO PADS 14 ns;

TIMESPEC TS_P2P = FROM PADS TO PADS 12 ns;

Vysvětlení:

Omezení P2F mezi vstupním pinem a D klopným obvodem nastaveno na 8 ns.

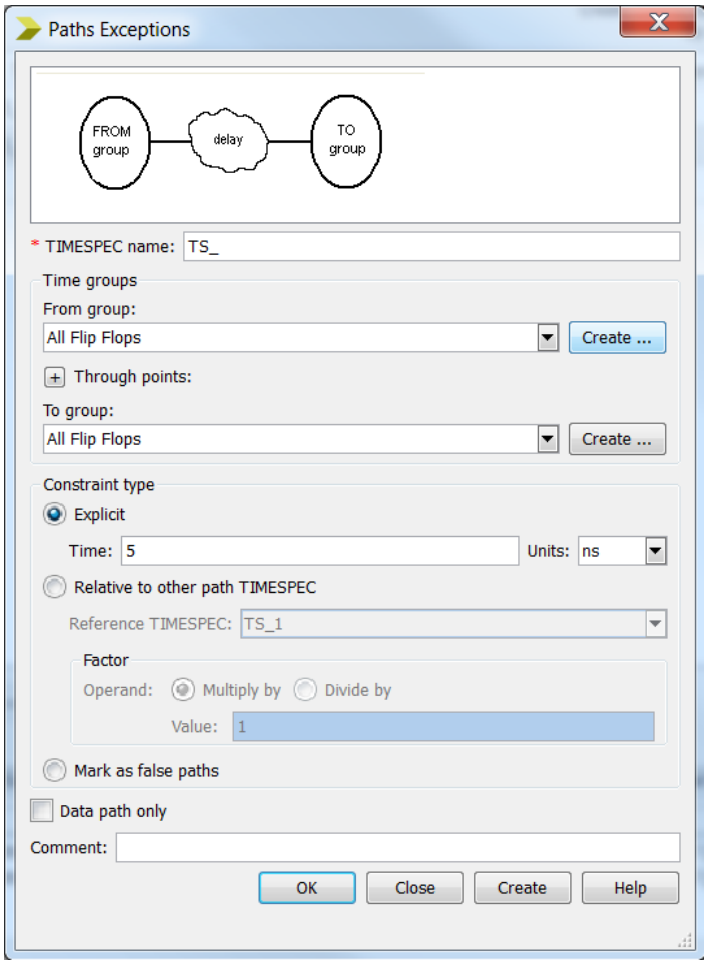
Omezení F2F mezi dvěma D klopnými obvody nastaveno na 5 ns.

Omezení F2P mezi D klopným obvodem a výstupním pinem nastaveno na 14 ns.

Omezení P2P mezi vstupně/výstupními piny nastaveno na 8 ns.

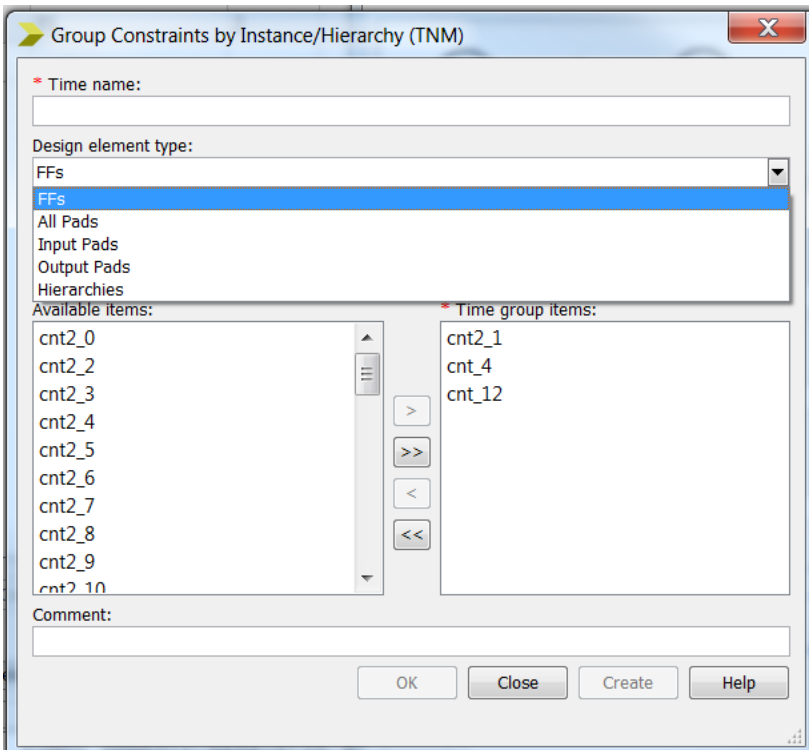
Nastavení hodnoty omezení FORM:TO pomocí rozhranní Create Timing Constraints v softwaru ISE Design Suite 14.7:

Začneme rozbalením záložky Timing Constraints, rozbalením Exceptions a pokračujeme dvojklikem na Paths, kde nastavíme hodnotu omezení, ale také mezi kterými z dostupných skupin má být omezení platné.



Obrázek 3 – nastavení omezení F2F na hodnotu 5 ns

Kliknutím na tlačítko Create u výběru skupiny se otevře nové okno s možností vytvořit si vlatní skupinu prvků u kterých chceme omezení FROM:TO nastavit.

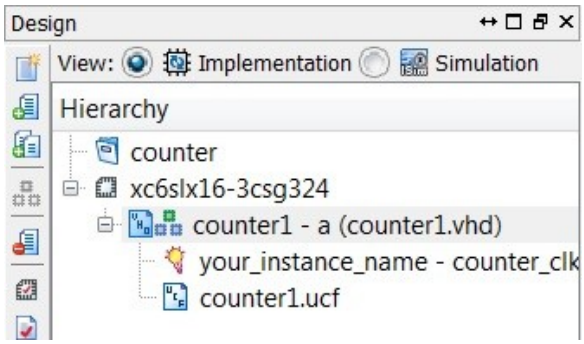


Obrázek 4 Tvorba vlastní skupiny vybraných prvků

2.3 DCM funkce CORE Generátoru

Je nutno také uvést obsluhu komponenty DCM (Digital Clock Manager), kde budeme nastavovat parametry hodinových signálů clk a clk1.

V okně Hierarchy 2xklikneme na položku „your_instance_name_counter_clk“



Obrázek 5 Zobrazení umístění komponentyDCM

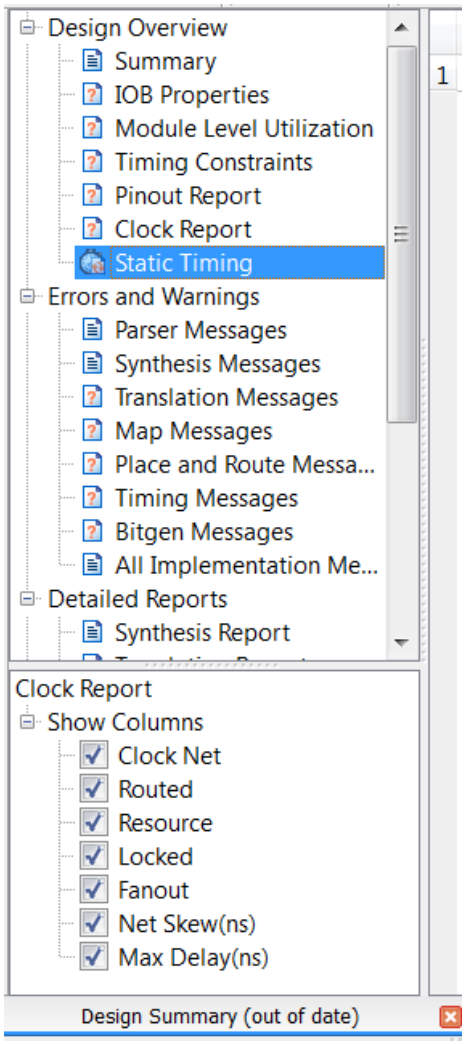
Zobrazí se nám okno s možností nastavení vstupní frekvence (Input Freq [MHz]) tedy frekvence signálu clk. Hodnota je nastavena na 100.000 tedy 100 MHz.

Kliknutím na tlačítko Next se posuneme na stranu 2, kde můžeme nastavit výstupní frekvenci, prozatím nastaveno na 100 MHz.

Po nastavení těchto dvou hodnot klikneme na Generate, čímž potvrdíme nové nastavení.

3. Zjištění časových parametrů

K zjištění časových parametrů budeme pracovat s nástrojem Static Timing, který najdeme v záložce Design Summary v kategorii Design Overview.



Obrázek 6 Zobrazení umístění Static Timing

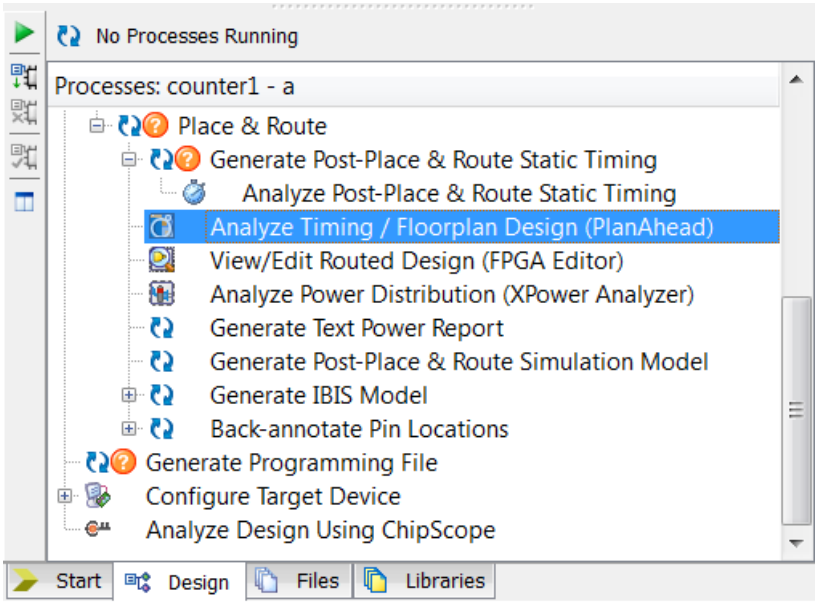
Po otevření nástroje Static Timing (který na liště najdeme pod položkou „counter1.twx“) můžeme v části Timing Summary najít klíčové parametry obvodu jako:

- Minimální periodu
- Maximální frekvenci
- Maximální zpoždění prvků omezení FROM/TO

V reportu nástroje Static Timing jsou uvedeny také všechny časové parametry zadaných omezení, i jednotlivých časy spojů mezi prvky.

4. Zobrazení rozvržení prvků na čipu FPGA

Pomocí nástroje PlanAhead, můžeme vidět využití jednotlivých buňek CLB buňek, vstupně/výstupních pinů či DCM bloků. Zkrátka si udělat představu jak se při určitém návrhu na FPGA čipu zapojila.



Obrázek 7 – Zobrazení kde najdeme nástroj PlanAhead

5. Popis funkce úlohy ve VHDL a na kitu NEXYS 3

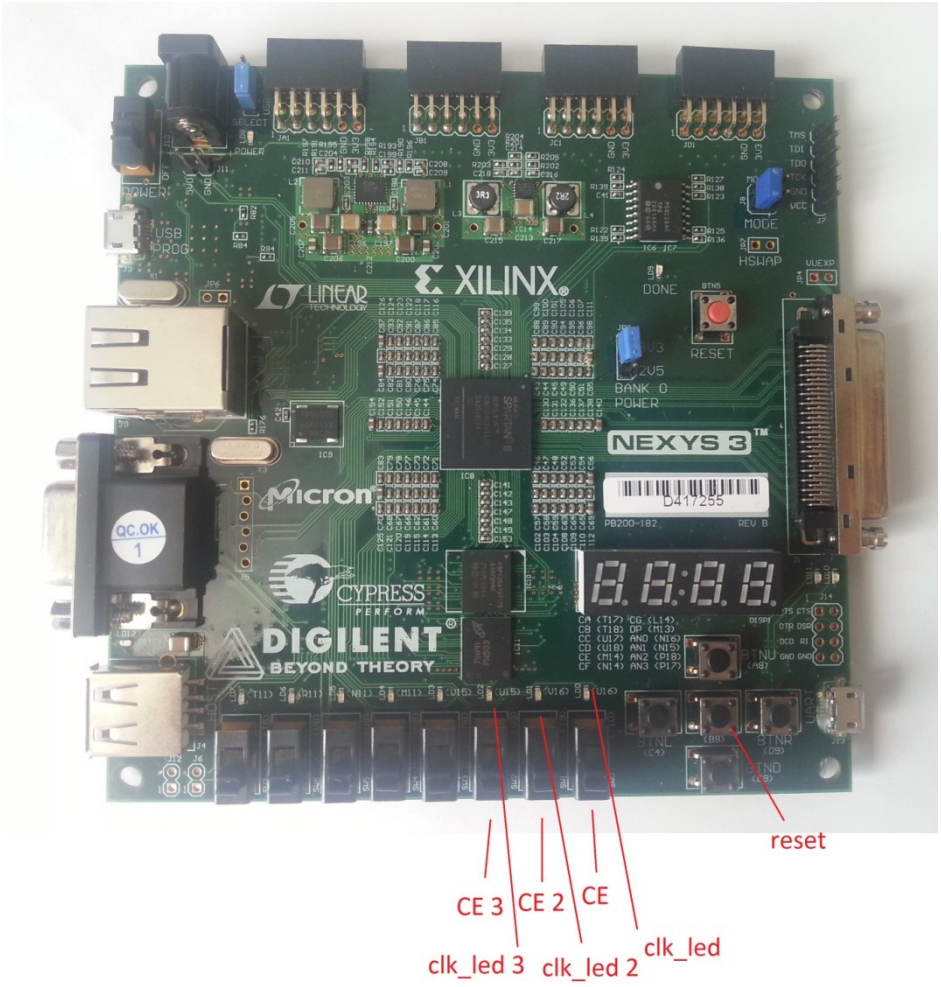
Funkce úlohy s názvem counter, tedy česky čítač, je prostá, jelikož v našem případě nejde o programování a následné vysvětlování složitého kódu ve VHDL, ale o demonstraci a pochopení problematiky časových omezení.

Ovládací a signalizační rozhraní:

Tabulka 1 Popis vstupů a výstupů

Vstupy	clk – základní hodinová frekvence krystalu 100 MHz
	CE – povolení hodin pro čitač 1
	CE2 – povolení hodin pro čitač 2
	CE3 – povolení hodin pro čitač 3

	reset – resetování čítačů
Výstupy	clk_led – signalizační led
	clk_led2 – signalizační led
	clk_led3 – signalizační led



Obrázek 8 Přiřazení rozhraní k ovládacím a signalizačním prvkům

Funkce:

Jedná se o tři čítače čítající od 0 do 16384 (rozsah zadán jen pro využití logiky) taktované výstupním signálem z DCM, které čítají při vzestupné hraně a povolených hodinách. Jakmile dočítají do konce, rozsvítí se signalizační led. Stisknutím tlačítka reset všechny tři vynulujeme. Tímto jsme schopni dokázat, že při rozsvícení led obvod funguje i při proměnných časových podmínkách, které budeme v průběhu měnit.

6. Zadání

1. Na začátku bez zadaných časových podmínek proveďte Implementaci a spusťte PlanAhead. Rovněž proveďte vygenerování bit souboru a funkci programu ověřte na desce NEXYS 3. Výsledek vložte do vypracování.
2. Ze statické časové analýzy uveďte data Timing Summary.
3. Pomocí omezení Period na hodinách clk zkuste dosáhnout co nejvyšší frekvence. Výsledek uveďte ve vypracování.
4. Nastavte Periodu hodin clk na 10 ns a zároveň pomocí DCM generátoru periodu clk1 na 400 MHz a opět proveďte implementaci a uveďte data Timing Sumarry. Zdali se objevily chyby ve spojích uveďte na kterých a proč.
5. Spusťte DCM generátor a změňte vstupní frekvenci na 200 MHz a výstupní ponechejte na 400 MHz. – navrhněte řešení vzniklého problému, je třeba si uvědomit vztah mezi chtěnou frekvencí a zadanou periodou, pomůže vám obsah varovných hlášení. Návrh řešení uveďte do vypracování.
6. Po opravení problému nastavte periodu na 5 ns (pokud jste již tak neudělali) spusťte Implementaci a uveďte hodnoty Timing Summary.
7. Obvod stále hlásí chybu, jedno řešení je uměle nastavit delší čas na pomalých spojích, navrhněte jak, možné řešení naleznete v kapitole **2 Možnosti omezení a nastavení parametrů hodinového signálu**. Implementujte a uveďte do vypracování: řešení a Timing summary.
8. Druhé řešení spočívá ve vyhovění podmínky na pomalých spojích. Přepočtěte zpoždění na frekvenci a tu zadejte do DMC generátoru jako výstupní frekvenci. Uveďte vypočtenou frekvenci a Timing Summary do vypracování.
9. Nyní když jste vyzkoušeli použití pár časových omezení, zkuste již bez návodu použít další a tím se pokusit návrh ještě více specifikovat.
10. Spusťte PlanAhead se zadanými omezeními a výsledek vložte do vypracování, proveďte rovněž vygenerování souboru bit. a funkci programu ověřte na desce NEXYS 3.
11. Zhodnoťte výsledky