Teoretický úvod – laboratorní úloha číslo 6

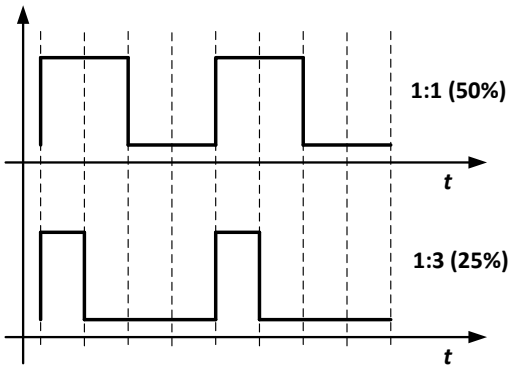
Děličky frekvence, digitální stopky s výstupem na displej

1 Děličky kmitočtu

Synchronní obvody vyžadují řízení pomocí vhodného signálu, který se obvykle označuje jako hodinový takt, ve VHDL se pro něj typicky používá pojmenování `Clock` či `Clk`. Hodinový signál je rovněž důležitý pro obsluhu a komunikaci s externími periferiemi připojenými k hradlovému poli FPGA. Většina přípravků, včetně desky Terasic DE10-Lite, však obsahuje obvykle pouze omezený počet zdrojů základního hodinového taktu, deska DE10-Lite je vybavena jedním oscilátorem, ze kterého hodinový obvod generuje a do FPGA pole dodává dvojici signálů s nominálními kmitočty 50 a 10 MHz. Zatímco pro komunikaci s jednotlivými prvky na desce přípravku (LCD displej, přepínače, otočný přepínač, tlačítka, porty rozhraní RS232) jsou potřeba různé taktovací a hodinové signály.

Tento problém se obvykle řeší pomocí realizace tzv. děliček kmitočtu, které umožňují odvodit ze základního hodinového signálu libovolný takt s požadovanou frekvencí. Alternativně výrobci FPGA přípravků často ke svým produktům nabízejí buď placené, či zdarma tzv. IP Cores (Intellectual Property), což jsou v podstatě SW i HW předpřipravené bloky různých základních funkcí, obvodů a částí, které lze využít ve vlastních projektech a při návrhu a realizaci vlastních obvodů.

Mezi těmito IP Cores bývají obvykle i obvody fázových závěsů (PLL) a bloků, které pracují s hodinovými signály, realizují děličky kmitočtů apod. V základním přiblížení představuje dělička kmitočtu ve své podstatě synchronní čítač, který čítá vzestupné (či naopak sestupné) hrany zdrojového hodinového signálu `Clock` a na svém výstupu produkuje výsledný hodinový takt v závislosti na požadovaném dělícím poměru, respektive délce použitého čítače. Modifikací základní děličky lze pak na výstupu získat signál s různým tvarem, tzv. střídou. Střída u periodických signálů představuje poměr mezi dobou trvání jeho jednotlivých stavů, tedy obvykle stavu logické jedničky a nuly, a udává se typicky jako poměr, např. 1:1 (neboli 50 %) představuje symetrický periodický signál se stejnou dobou trvání obou stavu logická 1 i 0, zatímco např. střída 1:3 (neboli 25 %) představuje délku trvání stavu logická 0 trojnásobnou oproti době trvání stavu logická 1. Obr. č. 1 znázorňuje uvedený rozdíl.



Obr. č. 1: Ilustrace rozdílu mezi signály s nesymetrickou a symetrickou střídou.

Záleží samozřejmě na konkrétní aplikaci, jakou střídu u výstupního signálu požadujeme, v praxi se ale obvykle využívá symetrická střída, zejména pro hodinové a taktovací signály. Děličky kmitočtu lze samozřejmě v jazyce VHDL realizovat a implementovat různými způsoby, zde uvedený je jen jeden z možných[1].

Častým způsobem realizace děličky kmitočtu je tzv. dělička typu 2^W, tedy dělička s poměrem úměrným mocnině čísla 2. Pro její vytvoření využijeme s výhodou datový typ `unsigned`, tedy převod logického vektoru na binární číselné vyjádření, definovaný v knihovně `numeric_std`. Děličku pak tvoří synchronní čítač, jehož stav vyjádříme typem `unsigned`. Při každé požadované hraně vstupního hodinového signálu (vzestupné či sestupné) přičteme do tohoto `unsigned +1`. Délku `unsigned` vektoru zvolíme v závislosti na požadovaném dělicím poměru děličky – nejvyšší bit vektoru `unsigned` se bude měnit mezi hodnotou logická 0 a logická 1 s periodou danou 2^{W-1}, a tedy perioda vzestupných (či naopak sestupných) hran nejvyššího bitu bude 2^W. Uveďme si tedy pro ukázku děličku 16 realizovanou výše uvedeným způsobem.

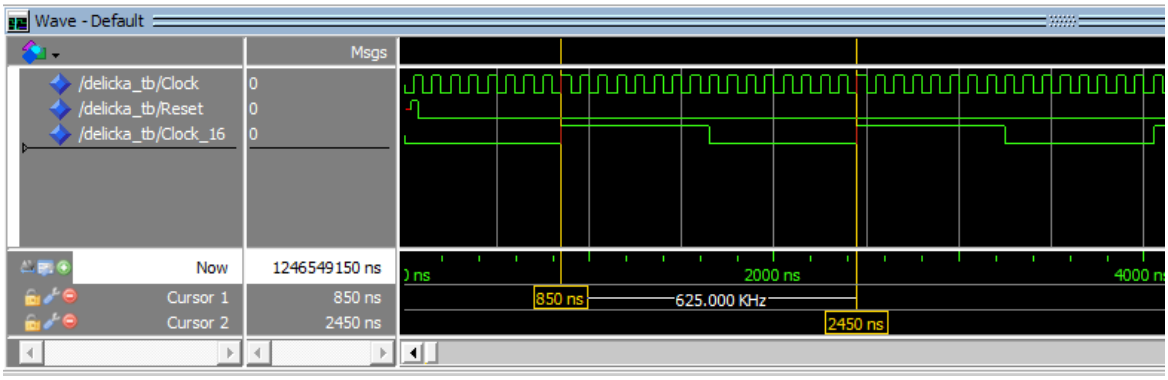
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity delicka is
port (Clock,Reset : in std_logic;
      Clock_16 : out std_logic);
end delicka;

architecture Behavioral of delicka is
begin
process(Clock,Reset)
variable citac : unsigned(3 downto 0):=(others=>'0');
begin
if Reset='1' then
     citac:=(others=>'0');
elsif Clock='1' and Clock'event then
     citac:=citac+1;
end if;
Clock_16<=std_logic(citac(3));
end process;
end Behavioral;
```

Navržená dělička obsahuje navíc asynchronní reset (nulování) se vstupem `Reset`. Vstupní hodinový signál je připojen na vstup `Clock`, výstupní hodinový signál po vydělení 16 je vyveden na výstup `Clock_16`. Jak vyplývá z označení tohoto typu děličky, 2^W, délku `unsgined` vektoru čítače v případě dělícího poměru 1:16 určíme jako 4 bity, tedy v jazyce VHDL `unsigned(3 downto 0)`. Nejvyšší bit čítače, `citac(3)`, pak po konverzi z datového typu `unsigned` na datový typ `std_logic` vysuneme do výstupu děličky[2].

[1] Podrobněji jsme se různými návrhy děliček kmitočtu zabývali na jedné z přednášek předmětu.

[2] Pokud bychom chtěli zobecnit uvedený řádek kódu, můžeme například použít atribut `'high`, který představuje nejvyšší bit vektoru. Uvedený zápis by pak v jazyce VHDL byl: `Clock_16<=std_logic(citac(citac'high));`.

Abychom ověřili funkčnost a správnost funkce navržené děličky, provedeme její simulaci v simulátoru Questa Intel FPGA. Jako vstupní hodinový signál použijeme takt 10 MHz, výstupem po vydělení 16 by tedy měl být hodinový signál s frekvencí 625 kHz. Obr. č. 2 zobrazuje časový výstup simulátoru.



Obr. č. 2: Simulace navržené děličky s dělícím poměrem 1:16.

Pomocí dvojice kurzorů změříme dobu uplynulou mezi dvojicí vzestupných hran výstupního signálu (portu) `Clock_16`, simulátor Questa uvedenou periodu převedl automaticky na vyjádření frekvence v Hz, výsledek 625 kHz odpovídá předpokladu a dělička kmitočtu s poměrem 1:16 tedy pracuje správně.

2 Dělička kmitočtu typu 2^W s parametrizací dělícího poměru

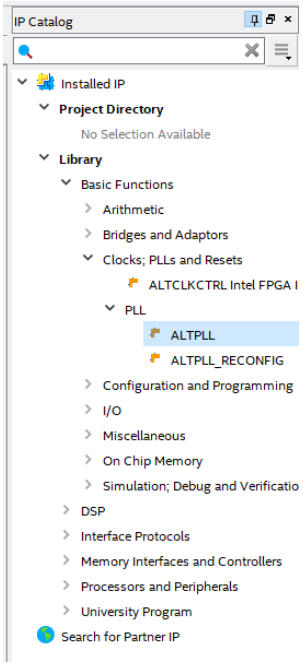
VHDL kód děličky s poměrem 1:16 a typu 2^W, který jsme si představili v předchozí kapitole, můžeme vhodně parametrizovat pomocí konstrukce `generic`, vytvořený návrh pak bude představovat obecnou děličku frekvence typu 2^W s dělícím poměrem daným parametrem `w`. Při využití dané děličky jako komponenty v rámci jiné top-level entity pak změnou tohoto parametru definujeme požadovaný dělící poměr jako 2^W. Výsledný VHDL kód parametrizované 2^W děličky tak můžeme zapsat:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity delicka is
generic (w : integer:=3);
port (Clock,Reset : in std_logic;
      Clock_2w : out std_logic);
end delicka;

architecture Behavioral of delicka is
begin
process(Clock,Reset)
variable citac : unsigned(w downto 0):=(others=>'0');
begin
if Reset='1' then
     citac:=(others=>'0');
elsif Clock='1' and Clock'event then
     citac:=citac+1;
end if;
Clock_2w<=std_logic(citac(w));
end process;
end Behavioral;
```

3 Využití IP Cores v programu Quartus, PLL fázový závěs

Výrobci FPGA přípravků a demonstračních kitů často ve svém programu pro jejich obsluhu zpřístupňují a vydávají tzv. IP Cores (Intellectual Property). Jedná se o předpřipravené a vygenerované VHDL kódy případně i bloky kódů s grafickou nadstavbou a ovládáním určené pro typické aplikace a pro využití různých vestavěných možností v rámci daného FPGA přípravku. Mohou být realizovány jak na HW tak i/nebo SW bázi, v některých programech mohou být tyto IP Cores placené, v jiných zdarma (nebo jsou zdarma jen některé základní, jiné za příplatek apod.).

V prostředí Quartus jsou k dispozici např. IP Cores pro aritmetické operace (násobičky, sčítačky…), základní logické a funkční bloky (registry, komparátory, kodéry a dekodéry,…), bloky inicializace a práce s pamětí a rovněž také blok fázového závěsu (PLL) pro práci a distribuci hodinových signálů, který využijeme v této laboratorní úloze. Obr. č. 3 ukazuje knihovnu všech dostupných IP Cores v prostředí Quartus; IP Catalog, nachází v pravém okně prostředí.



Obr. č. 3: Knihovna IP Cores v prostředí Quartus Lite.

Uvedený fázový závěs najdeme v knihovně pod záložkou Basic Functions → Clocks; PLLs and Resets → PLL a pro naši potřebu zvolíme typ ALTPLL. Nyní si představíme konfiguraci a inicializaci tohoto PLL fázového závěsu přesně tak, jak jej využijeme při návrhu a realizace digitálních stopek. V postupu práce se pak již na tuto konfiguraci jen odkážeme.

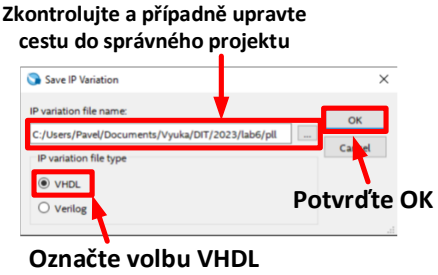
Fázový závěs PLL v rámci IP Core využijeme v této úloze pro realizaci prvního stupně dělení základního taktu a jeho vhodnou přípravu pro následující děličku typu 2^W, kterou pak již realizujeme čistě pomocí jazyka VHDL. Jako vstupní zdroj taktu pro celé digitální stopky využijeme oscilátor na přípravku s nominální frekvencí 10 MHz.

Protože navrhované stopky mají čítat v rozsah 0–59 sekund s rozlišením 1 sekundy, potřebujeme pro buzení čítače jednotek sekund hodinový signál s frekvencí 1 Hz. Integrovaný fázový závěs PLL nedokáže samostatně realizovat uvedený takt 1 Hz ze zdrojového 10 MHz, takový poměr je mimo rozsah integrovaného PLL, ale pomocí PLL můžeme vydělit zdrojový takt 10 MHz na takt nižší, jehož hodnota by byla rovna některé mocnině čísla 2 a ve druhém stupni dělení vytvořit v jazyce VHDL děličku typu 2^W, pomocí které získáme konečný takt 1 Hz.

Z možných hodnot vydělení zdrojového taktu v prvním stupni pomocí PLL závěsu se nabízí jako nejnižší možný takt 2048 Hz, což odpovídá hodnotě 2^{11}. Nakonfigurujeme proto ALTPLL modul pro realizaci výstupního hodinového signálu `Clock_2048` s frekvencí 2048 Hz a symetrickou střídou 1:1.

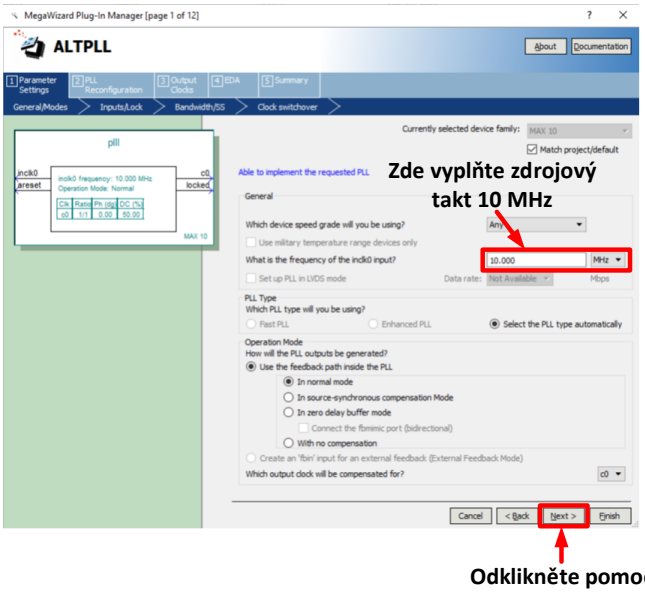
Obr. č. 3 ukazuje výběr z nabídky IP Cores modulu ALTPLL, v menu Basic Functions → Clocks; PLLs and Resets → PLL → ALTPLL; dále postupujeme podle návodu.

1. V prvním okně s názvem Save IP Variation zkontrolujeme cestu projektu digitálních stopek a zvolíme vhodný název výsledné entity, např. `pll`. Dále změníme volbu přepínače na „VHDL“ a potvrdíme pomocí OK.



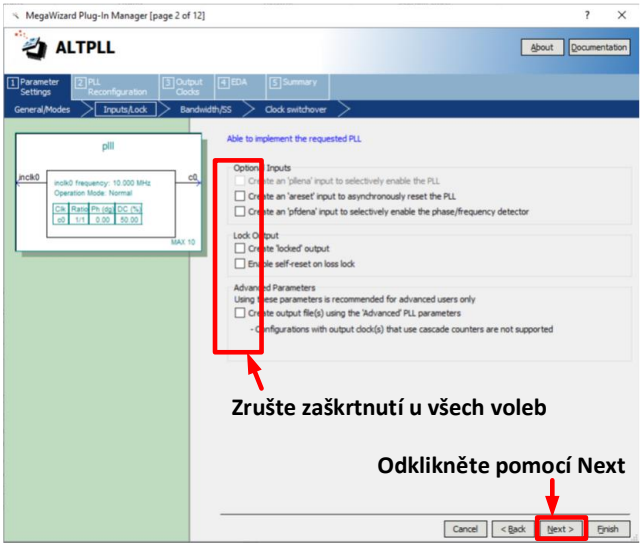
Obr. č. 4: Okno Save IP Variations.

2. Obr. č. 5 ukazuje další okno. Změňte jen hodnotu frekvence vstupního hodinového signálu na 10 MHz. Zbylé volby a možnosti ponechte a klikněte na tlačítko Next >.



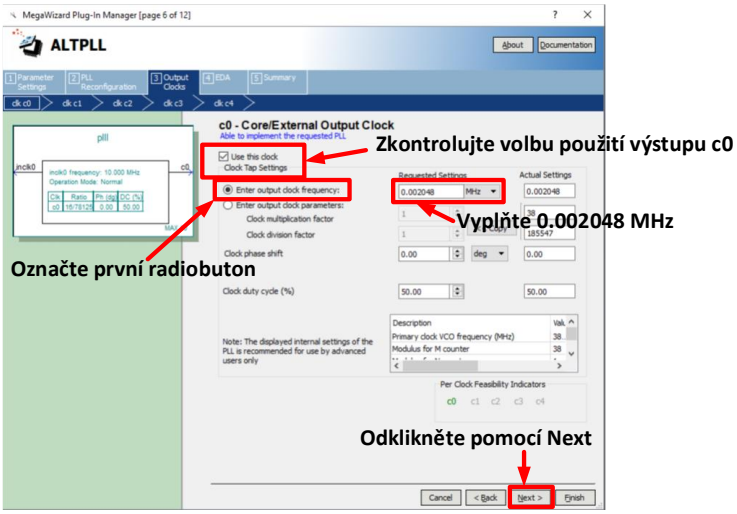
Obr. č. 5: Volba frekvence vstupního hodinového signálu.

3. Obr. č. 6: v následujícím okně odškrtněte (zrušte zaškrtnutí u všech voleb) a pokračujte opět Next >.



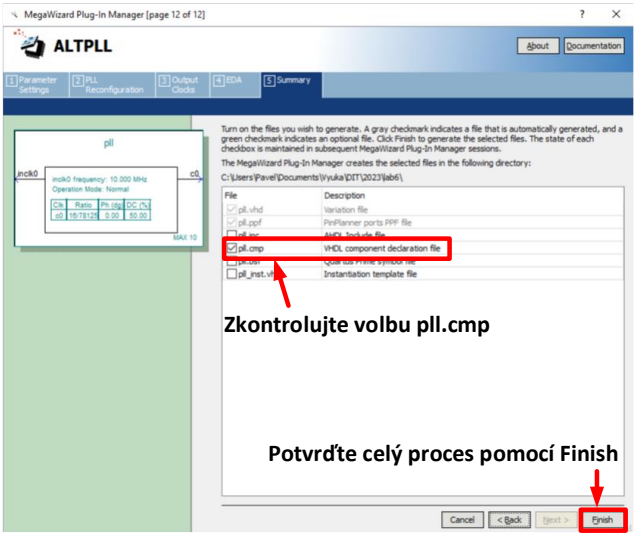
Obr. č. 6: Dodatečné volby PLL.

4. Okno č. 3 pouze odklikněte pomocí Next >, stejně tak i v okně č. 4 klikněte jen Next >, a opět v okně č. 5 klikněte pouze na Next >. Nyní v dalším okně, č. 6, je potřeba nastavit výstupní hodinový signál a jeho frekvenci. Nejprve zkontrolujte, že je zaškrtnuta volba „*Use this clock*“. Dále změňte stav přepínače pod ní na „*Enter output clock frequency*:“ a do zpřístupněného pole „*Requested Settings*“ vyplňte požadovanou frekvenci výstupního hodinového signálu 2048 Hz (tedy 0.002048 MHz), jak ukazuje další obrázek. Ostatní hodnoty ponechte beze změny a klikněte opět na Next >.



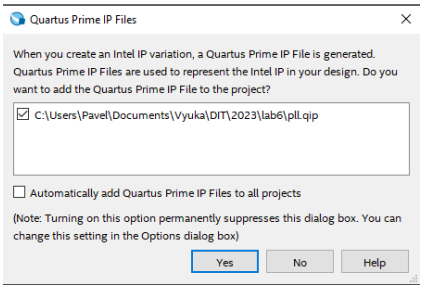
Obr. č. 7: Volba výstupu PLL fázového závěsu c0 a frekvence 2048 Hz.

5. Další okna č. 7, 8, 9 a 10 s volbami pro výstupy c1, c2, c3 a c4 pouze odklikejte pomocí Next >, stejně tak i následující okno č. 11 EDA pouze odklikněte prostřednictvím Next >. Obr. č. 8: v posledním okně č. 12 zkontrolujte, že je zaškrtnuta z nabídky výstupů pouze volba „*pll.cmp VHDL component declaration file*“. Klikněte na tlačítko *Finish*.



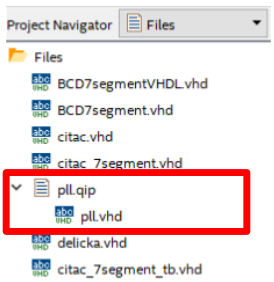
Obr. č. 8: Vložení komponenty pll do top-level entity a projektu.

6. Obr. č. 9: po dokončení průvodce vložení komponenty PLL, viz přechozí krok, se objeví okno s potvrzením vložení vytvořeného PLL fázového závěsu do aktuálního projektu. Potvrďte je pomocí *Yes*.



Obr. č. 9: Automatická volba pro vložení vytvořeného fázového závěsu PLL.

7. Obr. č. 10: Na závěr ještě zkontrolujeme, že vygenerovaný PLL modul byl úspěšně přidán do projektu. V hlavním okně programu Quartus zvolte v roletové nabídce v levé horní části „*Project Navigator*“ volbu „*Files*“ místo dosavadní „*Hierarchy*“. V zobrazeném seznamu souborů projektu by se měl nacházet soubor „*pll.qip*“ a u něho šipka pro rozbalení dalšího obsahu, ve kterém se nachází soubor „*pll.vhd*“.



Obr. č. 10: Kontrola vložení modulu PLL do projektu.

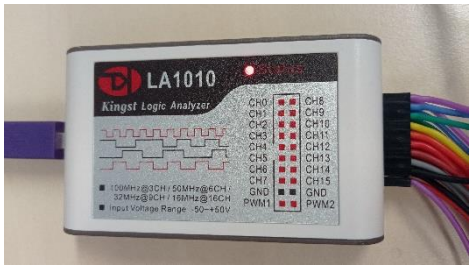
Pokud levým tlačítkem myši kliknete dvakrát na soubor *pll.vhd*, otevře se vygenerovaný VHDL kód komponenty PLL. Z něho pro nás bude důležitá zejména úvodní deklarace portů entity, kterou budeme potřebovat při vložení vytvořené komponenty v rámci top-level entity digitálních stopek.

```
ENTITY pll IS
     PORT
     (
          inclk0          : IN STD_LOGIC  := '0';
          c0         : OUT STD_LOGIC
     );
END pll;
```

4 Logický analyzátor Kingst 1010 a jeho připojení k přípravku DE10-Lite

Logický analyzátor je obecně zařízení určené k zachytávání a analýze digitálních (logických) signálů. Logické analyzátory existují jako čistě HW zařízení, kombinované HW/SW platformy i čistě SW řešení. Jsou k dispozici jako samostatná zařízení, ale často jsou rovněž integrována do digitálních osciloskopů jako jejich moduly. Analyzátor má obvykle větší počet vstupních kanálů, některé vstupy mohou být vyhrazeny pro synchronizaci, připojení země (GND) či napájení (VCC), generování signálů apod., často bývá omezen počet aktivních kanálů dle výše vzorkovací frekvence. Často obsahují funkce pro analýzu a dekódování řady různých komunikačních sběrnic a protokolů, např. 1-Wire, I2C, USB, RS232, UART, a řady dalších.

Obr. č. 11 ukazuje logický analyzátor Kingst LA1010 od společnosti Kingst Electronics, který budeme používat v laboratorní úloze č. 6.

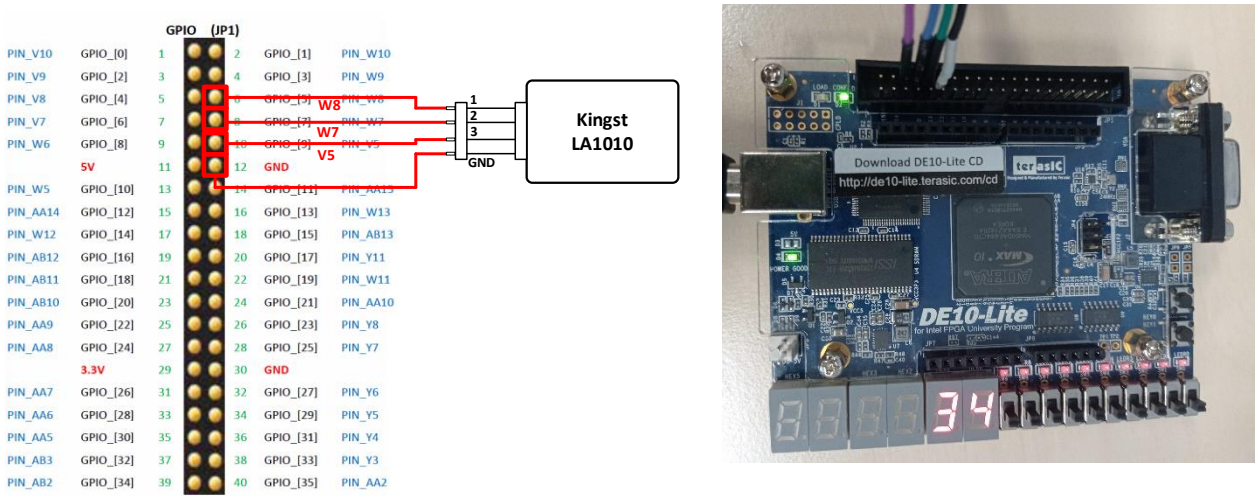


Obr. č. 11: Fotografie log. analyzátoru Kingst LA1010.

Log. analyzátor Kingst LA1010 je vybaven až 16 vstupy, vzorkovací frekvencí až 100 MHz, různými módy a režimy měření a USB rozhraním pro připojení k PC. Pomocí obslužného programu jsou na PC zobrazovány naměřené průběhy a program s nimi umožňuje různě pracovat a vyhodnocovat.

V této úloze si vyzkoušíme základní práci a obsluhu typického analyzátoru a pomocí typu Kingst LA1010 provedeme měření a ověření přesnosti hodinových taktů použitých v této úloze: 10 MHz, 2048 Hz a 1 Hz.

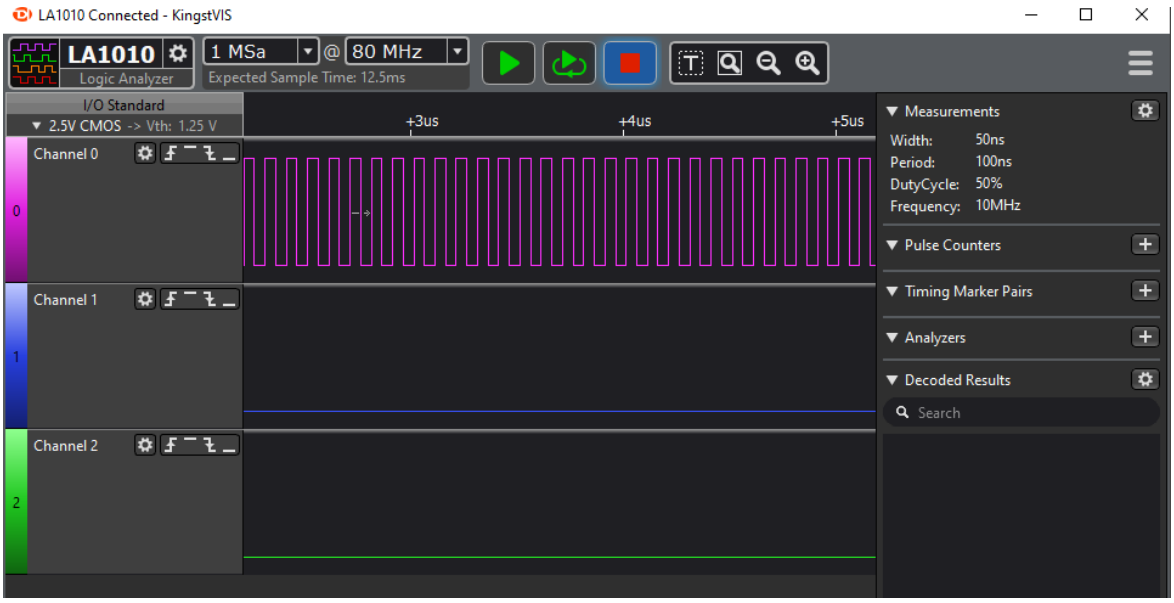
Obr. č. 12 znázorňuje připojení analyzátoru k přípravku DE10-Lite pomocí 3 kanálů a 1 uzemnění. Následně připojte log. analyzátor pomocí USB kabelu do volného USB portu Vašeho PC. V případě správného připojení analyzátoru k PC dojde k rozsvícení a pulzování červené LED diody na analyzátoru Kingst LA1010 označené *Status*.



Obr. č. 12: Připojení log. analyzátoru Kingst LA1010 k přípravku DE10-Lite.

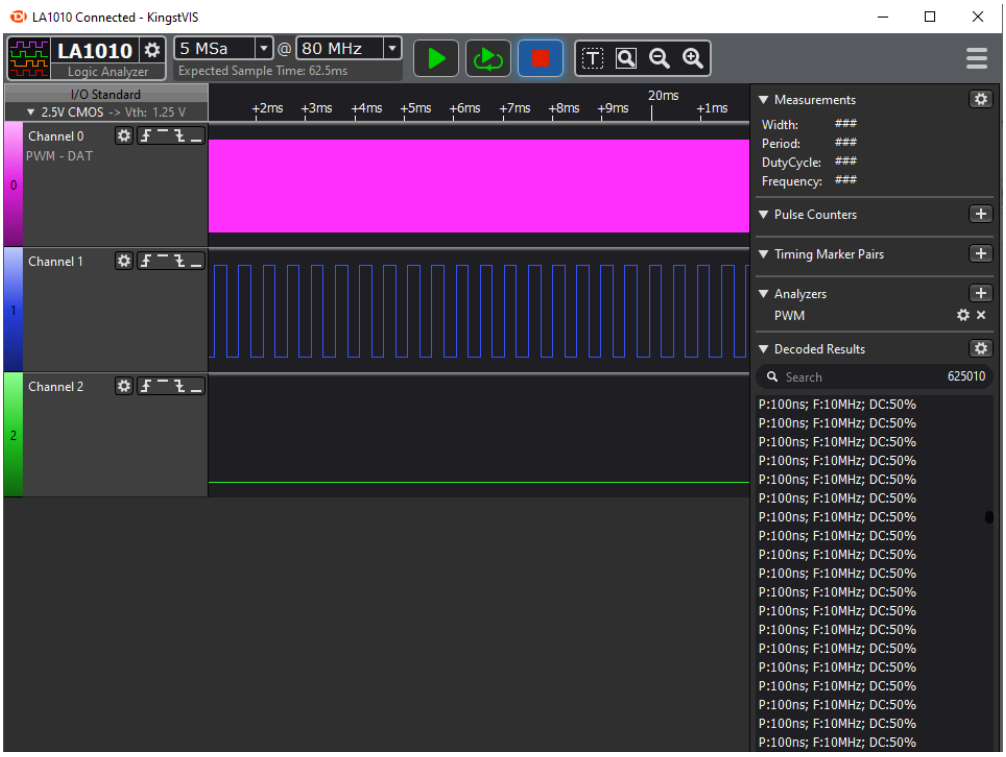
Nyní provedeme analýzu hodinových signálů přímo prostřednictvím logického analyzátoru, postupujte dle následujících kroků.

1. Na ploše PC klikněte a spusťte program pro obsluhu log. analyzátoru s názvem „*KingstVIS*“.
2. V horní hlavní liště programu „*I/O Standard*“ klikněte na symbol šipky a vyberte variantu „*2.5V CMOS*“.
3. Opět v horní liště nad volbou „*I/O Standards*“ klikněte na ikonu ozubeného kolečka vedle názvu analyzátoru „LA1010“. Tím se otevře základní nastavení analyzátoru a zobrazení programu KingstVIS. Zde v položce „*Channel Select*“ zaškrtněte zobrazení pouze prvních 3 kanálů, CH0, CH1 a CH2, u zbylých kanálů zaškrtnutí odeberte. Tím zůstanou v okně programu zobrazeny jen tyto 3 kanály.
4. Dále v horní liště zvolte v roletových menu vhodné nastavení vzorkovacího kmitočtu a počet vzorků pro správnou analýzu hodinových signálů vytvořených v této úloze. Protože na kanálu 0 (Channel 0) je připojen hodinový signál 10 MHz, na kanálu 1 (Channel 1) signál 2048 Hz a na kanálu 3 (Channel 3) pak signál 1 Hz, což představuje široký rozsah vstupních frekvencí, bude vhodné pro přesnou analýzu každého signálu zvolit jiné nastavení. (Nápověda – pro vzorkování 10 MHz signálu zvolte vzorkovací kmitočet 80 či 100 MHz, pro 2048 Hz zvolte 2 MHz a pro 1 Hz signál pak postačuje vzorkování pomocí 20 kHz, nastavení počtu vzorků ovlivní zejména dobu vzorkování.)
5. Spusťte vlastní vzorkování a analýzu pomocí symbolu zeleného trojúhelníku („*play*“) v horní liště programu. Počkejte dokončení vzorkování (kolečko „*Sampling*“ doběhne).
6. Najeďte kurzorem myši do zobrazeného navzorkovaného průběhu vybraného kanálu a pomocí kolečka myši můžete průběh přibližovat. Zobrazte si vhodně navzorkovaný průběh. Pokud budete kurzorem myši přejíždět po průběhu navzorkovaného signálu, bude se kurzor myši měnit dle šířky jednotlivých pulzů a v pravém sloupečku okna „*Measurements*“ programu KingstVIS vedle navzorkovaného průběhu se budou vypisovat aktuální hodnoty: šířka pulzu v ns, perioda průběhu v ns, střída signálu v % a z toho vypočítaná frekvence v Hz, jak ukazuje Obr. č. 13.



Obr. č. 13: Ukázka analýzy navzorkovaného signálu.

7. Dále v pravé části okna „*Measurements*“ klikněte na symbol plus „+“ v položce „*Analyzers*“. V nabídce vyberte možnost „*PWM*“. V otevřeném okně vyberte číslo kanálu, který chcete analyzovat, a dále pod touto nabídkou zaškrtněte volby „*Period*, *Frequency* a *Duty Cycle*“, zbylé volby ponechte odznačené. Klikněte *OK*.
8. Obr. č. 14: po této volbě se zobrazí v pravé části okna „*Measurements*“ pod položkou „*Decoded Results*“ výpis jednotlivých cyklů (period) daného hodinového signálu se zobrazením periody, frekvence a střídy.

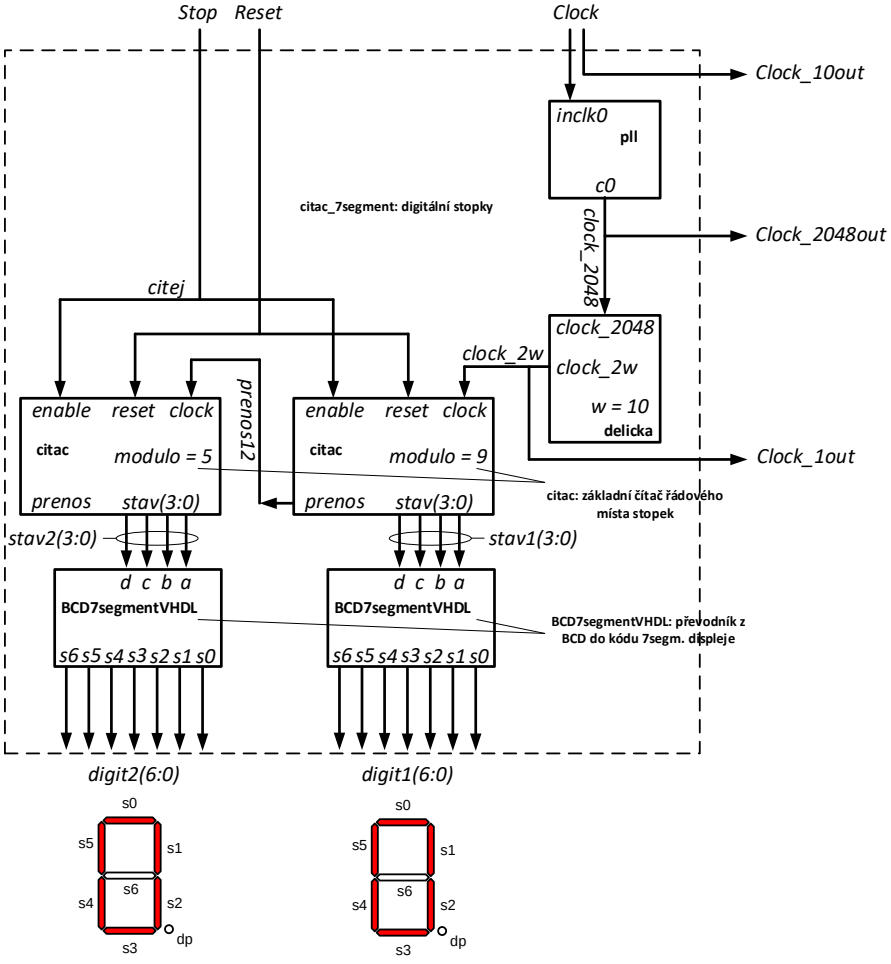


Obr. č. 14: Zobrazení navzorkovaných signálů a jejich analýza.

5 Realizace digitálních stopek

V předchozí laboratorní úloze jsme vytvořili jádro digitálních stopek i jejich hlavní části. V této úloze přidáme vhodné buzení hodinovým signálem a dále upravíme využití tlačítek (přidáme funkci pro zastavení a spouštění stopek); tím dokončíme jejich realizaci. Pomocí logického analyzátoru připojeného k trojici pinů přípravku s vyvedenými hodinovými signály, ověříme správnost a přesnost generovaných hodinových taktů: 10 MHz, 2048 Hz a 1 Hz.

Nakreslíme nejprve konečné schéma digitálních stopek a jejich strukturální zapojení pomocí jednotlivých komponent, signálů a portů. Obr. č. 15 ukazuje výsledek.



Obr. č. 15: Konečné schéma digitálních stopek a jejich komponent, signálů a portů.

Porovnáním s obdobným schématem základu stopek (čítače – stopek) z předchozí laboratorní úlohy je zřejmé, že změna nastala pouze v blocích pro hodinový vstup `Clock`. Zatímco v předchozí úloze jsme budili námi vytvořený čítač–stopky jednoduše stiskem tlačítka na přípravku, což nám posloužilo pouze jako kontrola správnosti a funkčnosti navržených čítačů a jejich výstupu na 7segmentové displeje, obsahuje řešení v této úloze skutečný zdroj hodinového signálu o frekvenci 1 Hz (signál `clock_2w`).

Ten je odvozen ze vstupu `Clock` pomocí dvojice frekvenčních děliček – nejprve v bloku `pll`, což je jeden z integrovaných fázových závěsů na přípravku a využitý pomocí IP Cores, dojde k vydělení vstupního hodinového signálu z portu `Clock` o frekvenci 10 MHz a přivedeného na vstup `inclk0` na výstupní hodinový signál na výstupu `c0` s frekvencí 2048 Hz.

Tento hodinový signál je pak prostřednictvím signálu `clock_2048` připojen na vstup `clock_2048` druhé frekvenční děličky s názvem `delicka` typu 2^W s parametrizací dělícího poměru `w`, kde dojde k jeho vydělení na výstupu `clock_2w` na výsledný hodinový signál o frekvenci 1 Hz. Ten je poté pomocí signálu `clock_2W` přiveden na vstup `clock` prvního čítače stopek, který čítá jednotky sekund.

Dále byly do návrhu výsledné entity digitálních stopek přidány výstupy `Clock_10out`, `Clock_2048out` a `Clock_1out`, na kterých jsou vyvedeny na externí piny přípravku jednotlivé hodinové signály 10 MHz, 2048 Hz a 1 Hz, a které slouží pro připojení logického analyzátoru Kingst LA1010, kterým ověříme správnost a přesnost generovaných hodinových signálů.

Další změnou realizace stopek oproti úloze č. 5 je využití tlačítka (KEY1) nově pro zastavení/spuštění stopek s využitím blokovacích vstupů čítačů `enable` prostřednictvím signálu `citej`.

Obr. č. 15 využijeme jako základ VHDL kódu pro celkovou realizaci uvedených digitálních stopek. Vyjdeme z VHDL kódu předchozí laboratorní úlohy č. 5, do které v podstatě jen doplníme dvojici nových komponent – `pll` a `delicka`. Realizaci PLL fázového závěsu jsme si podrobně vysvětlili v kapitole 3 a realizaci děličky typu 2^W v kapitole 2.

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

entity stopky is --deklarace portu entity stopek
port (Clock,Reset,Stop : in std_logic;
      Digit1,Digit2 : out std_logic_vector(0 to 6);
      Clock_10out,Clock_2048out,Clock_1out : out std_logic);
end stopky;

architecture Structural of stopky is
signal stav1,stav2: std_logic_vector(3 downto 0);
signal prenos12 : std_logic;
signal citej : std_logic :='1';
signal clock_2048,clock_2w : std_logic; -- signaly z/do delicek

component citac is -- komponenta zakladniho citace, viz Lab 5
...
end component;

component BCD7segmentVHDL is -- komponenta prevodniku z BCD kodu do kodu
7segmentoveho displeje, viz Lab 5
...
end component;

component delicka is -- komponenta delicky 2W
...
end component;

component pll is -- komponenta PLL fazoveho zavesu z IP Cores
...
end component;
```

```
begin
citac_1: citac -- mapovani zakladniho citace c. 1, viz Lab 5
generic map(...)
port map (...);

citac_2: citac -- mapovani zakladniho citace c. 2, viz Lab 5
generic map(...)
port map (...);

displej_1: BCD7segmentVHDL -- mapovani prevodniku do kodu 7segm. displeje
c. 1, viz Lab 5
port map (...);

displej_2: BCD7segmentVHDL -- mapovani prevodniku do kodu 7segm. displeje
c. 2, viz Lab 5
port map (...);

delicka_1: pll -- mapovani PLL fazoveho zavesu
port map (...);

delicka_2: delicka -- mapovani delicky typu 2W
generic map(...)
port map (...);

process(stop)
begin
... -- proces detekce stisknuti tlacitka spust/zastav stopky
end process;

Clock_10out<=Clock; -- vystup 10 MHz pro log. analyzator
Clock_2048out<=clock_2048; -- vystup 2048 Hz pro log. analyzator
Clock_1out<=clock_2w; -- vystup 1 Hz pro log. analyzator
end Structural;
```

V úvodní části entity jsme upravili porty – pro vstup `Stop` v rámci úlohy č. 6 budeme provádět přiřazení pinů, během přiřazení pinů dále upravíme přiřazení vstupu `Clock` (viz dále).

Pro připojení log. analyzátoru k přípravku a analýzu a kontrolu generovaných hodinových signálů vytvoříme 3 nové porty, `Clock_10out`, `Clock_2048out`, `Clock_1out`, pro vyvedení taktu 10 MHz, 2048 Hz a 1 Hz. Na začátku deklarace architektury definujeme nad rámec signálů použitých již v laboratorní úloze č. 5 signály použité pro propojení děliček kmitočtu, `clock_2048`, `clock_2w`, a tyto signály rovněž využijeme pro vyvedení hodinových taktů pro jejich analýzu pomocí log. analyzátoru Kingst LA1010.

Dále zde rovněž deklarujeme použití jednotlivých komponent ve finální top-level entitě stopek: `citac` – základní 4bitový synchronní čítač jako základ stopek, `BCD7segmentVHDL` – převodník z kódu BCD do kódu 7segmentového displeje pro zobrazení výstupu stopek na displeji, `delicka` – dělička frekvence typu 2^W ze vstupního hodinového taktu 2048 Hz na takt 1 Hz, `pll` – první stupeň dělení tvoří fázový závěs PLL vytvořený pomocí IP Cores a realizující výstupní takt 2048 Hz ze vstupního 10 MHz.

Obr. č. 15: po klíčovém slově `begin` následuje mapování portů (`port map`) a parametrů (`generic map`) jednotlivých komponent. Dále je v rámci architektury použit jeden proces, který při detekci sestupné hrany tlačítkového vstupu `Stop` provede inverzi pomocného signálu `citej`, který ovládá blokovací vstupy čítačů, a tedy provádí spuštění a zastavení digitálních stopek. Nakonec ještě v závěru architektury provedeme vyvedení jednotlivých taktů, 10 MHz, 2048 Hz a 1 Hz na výstupní porty, které slouží pro připojení log. analyzátoru Kingst LA1010.