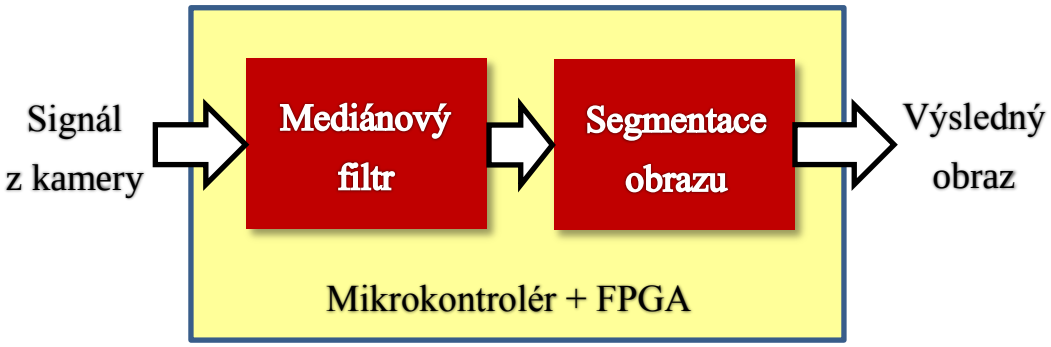
Vestavěný systém pro filtraci a segmentaci obrazu

Cílem projektu je vytvořit vestavěný systém, který s využitím kombinace mikrokontroléru a programovatelného obvodu typu FPGA vykonává filtraci a segmentaci obrazu z kamery (viz obrázek). Filtrace bude realizována skrze mediánový filtr. Segmentace obrazu pak bude implementována technikou prahování, kde hodnota prahu bude vypočtena z histogramu hodnot jednotlivých pixelů snímku metodou Otsu.



Jako vstup budete mít k dispozici kompletní algoritmus pro filtraci a segmentaci obrazu zapsaný v programovacím jazyce C, který lze zkompilovat a spustit na konvenčním procesoru. Vašim úkolem bude tento algoritmus přenést na platformu FITkit a vhodný způsobem jej rozdělit mezi dostupný mikrokontrolér MSP430 a FPGA čip s technologií SPARTAN3. S ohledem na omezené množství výpočetních zdrojů na platformě FITkit je požadováno, aby výsledná aplikace byla schopna zpracovat obraz s rozlišením 320x240 pixelů, který bude generován s rychlostí 60 snímků za vteřinu (více informací o omezeních platformy FITkit bude uvedeno níže).

Postup práce

1. V informačního systému si z adresáře *Projekty* stáhněte archiv s názvem *projekt.zip* a seznamte se s jeho obsahem (viz kapitola ***Struktura vstupních souborů***).
2. Seznamte se vstupním algoritmem pro filtraci a segmentaci obrazu, který je umístěn v souborech *cpu/cpu.c* a *cpu/program.c*. Pozornost zaměřte na pochopení principu filtrace obrazu s použitím mediánového filtru, způsobu ***Zpracování vstupních pixelů*** a segmentace obrazu s použitím prahování, kde je práh vypočten dynamicky na základě histogramu metodou Otsu (viz samostatný dokument v IS). Všimněte si prosím, že vstupní obraz je v souboru generován s rozlišením 320x240 pixelů, kde každý pixel reprezentuje jeden z osmi možných odstínů šedi. Toto omezení vyplývá z vlastností přípravku FITkit (více informací bude uvedeno níže), které však nemají vliv na obecnost této úlohy. Dále si všimněte, že vstupní obraz je v algoritmu generován a parametrizován na základě loginu – **jeho hodnotu prosím změňte na Váš vlastní login v souboru *cpu/common.h***. Na standardní výstup vypisuje program hodnoty histogramů vybraných 10 snímků a také vypočtené hodnoty prahu metodou Otsu. Konkrétněji, program počítá histogram a novou hodnotu prahu u každého desátého snímku a tyto informace zobrazuje jednou za 100 snímků. Pozn.: pokud nepoužíváte na svém osobním počítači operační systém typu linux, můžete pro kompilaci a spuštění využít např. stroj *merlin.fit.vutbr.cz*.
3. Vytvořený algoritmus přeneste na mikrotontrolér MSP430 (do souboru *mcu/main_sw.c*) a vytvořte tak čistě softwarovou implementaci algoritmu (bez účasti FPGA). Ověřte funkčnost této implementace přímo na přípravku FITkit (postupujte dle návodu uvedeného v kapitole ***Ověření aplikace na přípravku FITkit***). **Výstup generovaný na terminálu FITkitu musí být stejný jako výstup generovaný programem *cpu/program.c*. *Důležitá poznámka:*** Doba zpracování jednotlivých pixelů je na mikrokontroléru výrazně delší než na CPU. Aby jste se dočkali výsledků v rozumném čase, je pro vás připravena upravená funkce generátoru pixelů, která dokáže posouvat generátor o několik snímků vpřed. Základní myšlenkou pro získání stejných hodnot histogramů jako na CPU je spustit výpočet na daném snímku a následně posunout generátor o 99 snímků vpřed, spočítat další snímek, posunout generátor o 99 snímků, atd. Na závěr tohoto bodu změřte

průměrnou dobu zpracování jednoho pixelu snímku. Pro změření tohoto času využijte předpřipravenou funkci *get_time()*. Přepínač *PROFILE* využijte pro výběr mezi režimem měření času a režimem výpisu histogramů a vypočteného prahu na terminál.

4. Z předchozího bodu by Vám mělo vyplynout, že samotný mikrokontrolér MSP430 nemá dostatečný výkon pro zpracování obrazu s rozlišením 320x240 pixelů, 60 snímků za vteřinu. Analyzujte proto časově kritické části algoritmu s využitím volně dostupného programu *gprof*, který je schopen změřit procentuální zastoupení jednotlivých funkcí algoritmu v rámci přiděleného procesorového času. Pro získání přesnějších údajů proveďte měření alespoň desetkrát a použijte průměrnou hodnotu z naměřených výsledků. Jelikož program *gprof* není snadné aplikovat přímo na mikrokontroléru MSP430, proveďte tuto profilaci s původním algoritmem v souboru *cpu/program.c* a na běžném procesoru. Pozn.: pokud nepoužíváte na svém osobním počítači operační systém typu linux, můžete pro profilaci použít např. stroj *merlin.fit.vutbr.cz*. Při profilaci aktivujte přepínač *PROFILE*, který zajistí, že se do naměřených hodnot nebude započítávat tisk histogramů a vypočtených prahů na standardní výstup.
5. Seznamte se s ***Architektura systému na přípravku FITkit*** a ověřte si funkčnost předpřipravené demo aplikace pro komunikaci mezi mikrokontrolérem a FPGA čipem. Pro vyzkoušení této demo aplikace postačí změnit v souboru *project.xml* zdrojový kód pro mikrokontrolér z *mcu/main_sw.c* na *mcu/main_swhw.c* a opět zkompilovat a naprogramovat přípravek FITkit. Jakmile se Vám podaří úspěšný test této demo aplikace, vyzkoušejte si dále postup, jakým lze pomocí nástroje Catapult C získat ze zdrojových souborů v jazyce C výstupní soubory v jazyce VHDL. Postupujte dle návodu v kapitole ***Syntéza souborů s využitím nástroje Catapult C*** a aplikujte jej na soubor *fpga/src_filter/filter.cpp*. Na závěr si tímto postupem vygenerujte také nový VHDL soubor pro generátor pixelů (soubor *fpga/src_genpix/genpix.cpp*), který bude využívat **Vámi nastavený login ze soboru *cpu/common.h***.
6. Na základě informací z bodů 3 a 4 identifikujte ve vstupním algoritmu části, které je vhodné ponechat na mikrokontroléru a části, které je nezbytné umístit do FPGA. Současně s rozdělením aplikace na hardwarovou a softwarovou část navrhněte vhodný způsob předávání dat mezi mikrokontrolérem a FPGA čipem. Proveďte implementaci navrženého systému a jeho **funkčnost ověřte nejprve simulací** v prostředí Catapult C. Pro účely simulace je pro Vás připraven testbench soubor *fpga/src_filter/tb_filter.cpp*, který doplňte tak, aby simuloval část aplikace běžící na mikrokontroléru. Jakmile bude simulace vykazovat bezchybné výsledky, přistupte k ověření aplikace na přípravku FITkit. **Zkontrolujte, zda výstup generovaný na terminálu FITkitu se shoduje s výstupem generovaným původním programem *cpu/program.c*.** Pozn.: Při implementaci prosím dbejte na efektivní využití zdrojů v FPGA čipu a jednotlivé datové typy proměnných optimalizujte na nezbytně nutnou datovou šířku. Ve vstupním zdrojovém kódu je pro vás připravena modifikovaná verze funkce pro výpočet mediánu, která je vhodnější pro hardwarovou realizaci.
7. Vytvořte technickou zprávu přesně dle pokynů uvedených v kapitole ***Pokyny pro vypracování technické zprávy***.

Výstupy projektu

Prostřednictvím informačního systému budete odevzdávat archiv *projekt.zip*, který musí obsahovat následující soubory:

- Implementace aplikace běžící pouze na mikrokontroléru (bez FPGA)
  - `mcu/main_sw.c` – kód pro MSP430
- Implementace aplikace rozdělené mezi HW a SW
  - `mcu/main_swhw.c` – kód pro MSP430
  - `fpga/src_filter/filter.cpp, filter.h, addr_space.h` – kód pro FPGA
  - `fpga/src_filter/directives.tcl` – skript, kterým byla provedena syntéza souboru `filter.cpp` do jazyka VHDL
  - `fpga/src_filter/tb_filter.cpp` – doplněný testbench soubor, který slouží pro porovnání výstupu původního a modifikovaného algoritmu.
  - `fpga/src_filter/mapped.vhdl` – upravený výstupní soubor filtru z prostředí Catapult C, který lze přímo přeložit v aplikaci QDevKit
- Technická zpráva vypracovaná přesně dle pokynů
  - `report.pdf`

Pozor, v případě opisování budou pachatelé předvedeni před disciplinární komisi a projekt bude hodnocen nula body!

Doplňující informace

Struktura vstupních souborů

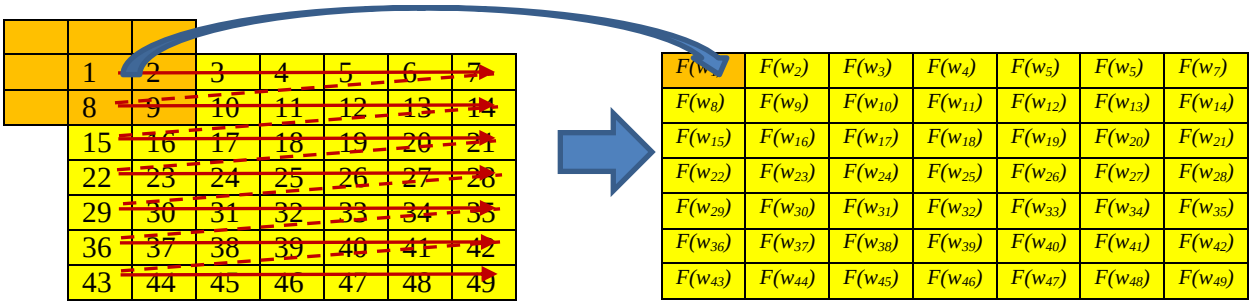
Archiv *projekt.zip* obsahuje adresář s názvem *segmentace* obsahující následující části:

- Vstupní algoritmus pro filtraci a segmentaci obrazu určený pro běžný procesor
  - `cpu/` – zdrojové kódy a Makefile
- Předpřipravený soubor pro aplikaci běžící pouze na mikrokontroléru (bez FPGA)
  - `mcu/main_sw.c` – kód pro MSP430
- Ukázková aplikace pro komunikaci mezi mikrokontrolérem a FPGA, která současně slouží jako šablona pro výsledné řešení
  - `mcu/main_swhw.c` – kód pro MSP430
  - `fpga/src_filter/` – zdrojový kód, testbench soubor a překladové skripty pro filtr
  - `fpga/src_genpix/` – zdrojový kód, testbench soubor a překladové skripty pro generátor pixelů
  - `fpga/src_video/` – zdrojový kód, testbench soubor a překladové skripty pro video buffer
  - `fpga/top_level.vhd` – VHDL soubor propojující všechny komponenty uvnitř FPGA
  - `fpga/sim/` – testbench soubor a fdo skript pro ModelSIM umožňující spustit simulaci všech komponent v FPGA
- Konfigurační soubor pro překlad a spuštění aplikace v prostředí QDevKit
  - `project.xml`
- Pomocné soubory
  - `convert.bat` – zajistí konverzi výstupního VHDL souboru z nástroje Catapult C do podoby, která je přeložitelná v nástroji Xilinx ISE.

Více informací o úloze jednotlivých komponent v FPGA a jejich propojení naleznete níže v kapitole ***Architektura systému na přípravku FITkit***.

Zpracování vstupních pixelů

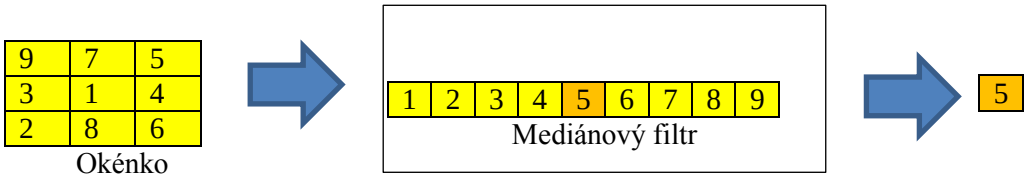
Celá řada obrazových operátorů v počítačové grafice používá posuvné okénko 3x3 pixelů, kterým postupně prochází celý snímek a na jejímž základě vypočítává hodnoty pixelů nového snímku (viz obrázek).



Původní snímek

Nový snímek

Mezi tyto operátory patří i mediánový filtr, který nové hodnoty pixelů (vyfiltrovaného snímku) vypočítává na základě okolních bodu původního snímku. Samotnou funkci mediánového filtru *F(w)* lze popsat následovně: filtr hodnoty pixelů v posuvném okénku nejprve seřadí, a potom na svůj výstup vydá prostřední hodnotu této seřazené posloupnosti (viz obrázek).



Okénko

Mediánový filtr

Při aplikaci posuvného okénka vzniká problém, jak se zachovat k pixelům na okrajích snímku, ke kterým nemáme k dispozici okolní body. Jedním z nejjednodušších způsobů řešení tohoto problému je replikace okrajových bodů (někdy označovaná také jako *window clipping*). Názorně je tato operace zobrazena na následujícím obrázku. Na takto rozšířený snímek již lze snadno aplikovat posuvné okénko i pro okrajové body. Samotné rozšíření snímku je možné realizovat buď *přímo* - skutečným zvětšením snímku; nebo *nepřímo* (až v průběhu výpočtu) – velikost snímku zůstává stejná, avšak v okamžiku zpracování některého z okrajových pixelů je okénko 3x3 vytvořeno replikací vnitřních bodů. V praktických aplikacích je častěji využíván tento druhý způsob, který nevyžaduje skutečné zvýšení rozlišení snímku.

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49

Původní snímek

1	1	2	3	4	5	6	7	7
1	1	2	3	4	5	6	7	7
8	8	9	10	11	12	13	14	14
15	15	16	17	18	19	20	21	21
22	22	23	24	25	26	27	28	28
29	29	30	31	32	33	34	35	35
36	36	37	38	39	40	41	42	42
43	43	44	45	46	47	48	49	49
43	43	44	45	46	47	48	49	49

Snímek rozšířený o okrajové pixely

Pokud je snímek zpracováván na běžném počítači, je obvykle k dispozici dostatek paměti pro jeho uložení. Pokud je však zpracování obrazu přesunuto na některé vestavěné zařízení nebo přímo do specializovaného čipu, potom je nezbytné s paměťovým prostorem co nejvíce šetřit. Vstupní obraz je do těchto zařízení zasílán obvykle ve formě sériového proudu dat (pixel po pixelu). Pokud si uvědomíme, jakým způsobem je posuvné okénko postupně aplikováno na celý snímek, zjistíme, že pro zpracování libovolného pixelu nám postačí paměť o velikosti nejvýše dvou řádků a tří pixelů (viz obrázek).

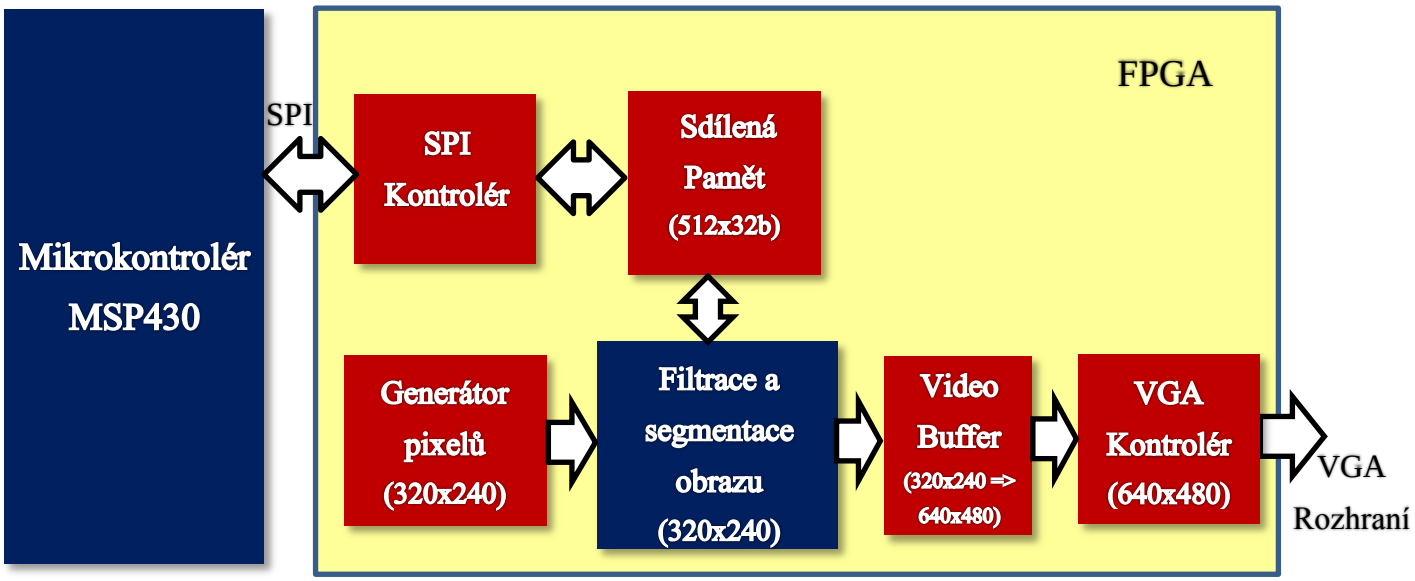
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49

Zvýrazněné pixely označují oblast, která musí být uložena v pomocné paměti

Architektura systému na přípravku FITkit

Předpřipravená architektura systému pro filtraci a segmentaci obrazu na přípravku FITkit je znázorněna na následujícím obrázku. Tento systém zahrnuje mikrokontrolér MSP430 a FPGA čip s technologií SPARTAN3, které jsou propojeny skrze standardní sériové rozhraní SPI. Propustnost tohoto rozhraní je pouze 460 kB/s a je navíc sdílena pro oba směry komunikace. Data jsou mezi mikrokontrolérem a FPGA čipem vyměňována skrze sdílenou paměť o velikosti 512x32b, která je umístěna na FPGA čipu (realizováno jako dvou-portová paměť typu BlockRAM). **Obsah této sdílené paměti je po spuštění aplikace vynulován.** Zatímco mikrokontrolér k obsahu této sdílené paměti přistupuje skrze SPI rozhraní a SPI kontrolér, FPGA čip může přistupovat přímo. V ukázkovém souboru pro mikrokontrolér jsou pro Vás připraveny pomocné funkce *fpga_read(addr)* a *fpga_write(addr, data)*, skrze které lze snadno z této paměti číst nebo do ní zapisovat.

Přípravek FITkit nemá bohužel k dispozici rozhraní pro vstup video signálu z kamery, a proto musí být tento vstup generován uvnitř systému. Jelikož je propustnost mezi mikrokontrolérem a FPGA čipem příliš nízká, není možné pixely generovat v mikrokontroléru a přenášet do FPGA a musí tak být generovány přímo uvnitř FPGA (komponenta *Generátor pixelů*). Jednotlivé pixely jsou z generátoru zasílány komponentě *Filtrace a segmentace obrazu*, která je ústřední částí celé architektury. Jednotlivé pixely vyfiltrovaného a segmentovaného obrazu jsou následně zasílány do *Video bufferu* a *VGA kontroléru.* Tyto komponenty zajišťují správné zobrazení výsledného snímku na VGA konektoru přípravku FITkit. Korektní chování Vaší aplikace bude tak možné ověřil na kterémkoliv monitoru podporujícím režim 640x480, 60 snímků za vteřinu.



Přípravek FITkit obsahuje několik omezení, které je potřeba vzít při řešení této úlohy v úvahu. Mezi tato omezení patří:

- Počet odstínů zobrazovaných na VGA rozhraní je pouze 8. Přesněji řečeno, pro každou barevnou složku (RGB) jsou na VGA rozhraní přípravku rezervovány pouze 3 bity. Jelikož tento projekt pracuje pouze s odstíny šedi, je k dispozici pouze 8 těchto odstínů.
- S ohledem na omezené výpočetní zdroje přípravku FITkit pracuje *Generátor pixelů* a obvod pro *Filtraci a segmentaci obrazu* pouze s polovičním rozlišením obrazu tj. 320x240, 60 snímků za vteřinu. Teprve až komponenta *Video buffer* provede transformaci tohoto obrazu na rozlišení 640x480 tak, aby jej bylo možné sledovat na monitoru. Pozn.: *VGA Kontrolér* nepodporuje nižší rozlišení než 640x480 pixelů.
- Obvod umístěný v FPGA pracuje na frekvenci 25MHz, která odpovídá frekvenci, s jakou jsou generovány jednotlivé pixely na VGA rozhraní s rozlišením 640x480. Jelikož filtrace a segmentace obrazu pracuje s polovičním rozlišením 320x240, je počet pixelů čtvrtinový oproti rozlišení 640x480 (320x240 = 640x480/4). Jednotka pro filtraci a segmentaci má proto k dispozici nejvýše 4 takty hodinového signálu mezi zpracováním jednotlivých pixelů. Jinými slovy každé 4 takty hodinového signálu musí začít zpracování nového pixelu (tzv. inicializační interval smyčky), délka tohoto zpracování (tj. latence) však může být libovolná.
- Výpočet histogramu a odvození nového prahu metodou Otsu se provede pouze jednou za 10 snímků obrazu (častější aktualizace není potřeba). Jelikož výpočet nového prahu metodou Otsu trvá určitou dobu, není možné jej aplikovat ihned na následující snímek obrazu, ale až o snímek později. Jinými slovy, pokud je vypočten histogram ze snímku s indexem *i*, potom v průběhu snímku *i+1* je vypočtena nová hodnota prahu a tuto novou hodnotu lze aplikovat až od prvního pixelu snímku *i+2*.

V průběhu řešení tohoto projektu budete primárně modifikovat kód mikrokontroléru a komponenty *Filtrace a segmentace obrazu*. Pro správnou implementaci komponenty *Filtrace a segmentace obrazu* je potřeba pochopit rozhraní této komponenty jak na úrovni jazyka C, tak na úrovni výsledného obvodu v jazyce VHDL.

Rozhraní na úrovni jazyka C je následující:

Název	Typ	Orientace	Popis
in_data	t_pixel	Vstup	Vstupní pixel
in_data_vld	bool	Vstup	Informace o tom, zda je vstupní pixel platný
out_data	t_pixel	Výstup	Výstupní pixel
mcu_data	t_mcu_data[512]	Vstup/Výstup	Sdílená paměť pro výměnu dat s MCU

Na úrovni výsledného obvodu v jazyce VHDL je potřeba, aby se jednotlivé parametry chovaly následovně:

Název	Rozhraní - Catapult	Rozhraní - VHDL
in_data	mgc_in_wire_en	in_data_rsc_z : IN std_logic_vector (2 DOWNTO 0); in_data_rsc_lz : OUT std_logic;
Pozn.: tento typ rozhraní přidá k samotné datové sběrnici *in_data* i jednobitový signál (*_lz*), jímž si komponenta žádá o nová data. Jinými slovy, tento signál je aktivován, kdykoliv je uvnitř kódu v jazyce C přečtena proměnná *in_data*.		
in_data_vld	mgc_in_wire	in_data_vld_rsc_z : IN std_logic;
Pozn.: jedná se pouze o jednobitový vodič informující o platnosti dat.		
out_data	mgc_out_stdreg_en	out_data_rsc_z : OUT std_logic_vector (2 DOWNTO 0); out_data_rsc_lz : OUT std_logic ;
Pozn.: Tento typ rozhraní přidá k samotné datové sběrnici *out_data* i jednobitový signál (*_lz*), jímž komponenta potvrzuje platnost výstupních dat. Jinými slovy, tento signál je aktivován, kdykoliv je uvnitř kódu v jazyce C zapsána hodnota do proměnné *out_data*.		
mcu_data	singleport RAM	mcu_data_rsc_singleport_data_in : OUT std_logic_vector (31 DOWNTO 0); mcu_data_rsc_singleport_addr : OUT std_logic_vector (8 DOWNTO 0); mcu_data_rsc_singleport_re : OUT std_logic; mcu_data_rsc_singleport_we : OUT std_logic; mcu_data_rsc_singleport_data_out : IN std_logic_vector (31 DOWNTO 0);
Pozn.: Tento typ rozhraní znamená, že uvedené pole bude umístěno v externí paměti typu RAM a rozhraní komponenty bude doplněno o běžné paměťové rozhraní *(_data_in, _addr, _re, _we, _data_out*). Kdykoliv je uvnitř kódu v jazyce C čtena nebo zapisována položka z/do tohoto pole, je vygenerována příslušná čtecí nebo zápisová transakce na tomto rozhraní.		

Ověření aplikace na přípravku FITkit

Vývojové nástroje

Pro vypracování projektu a jeho ověření na přípravku FITkit jsou potřeba následující nástroje:

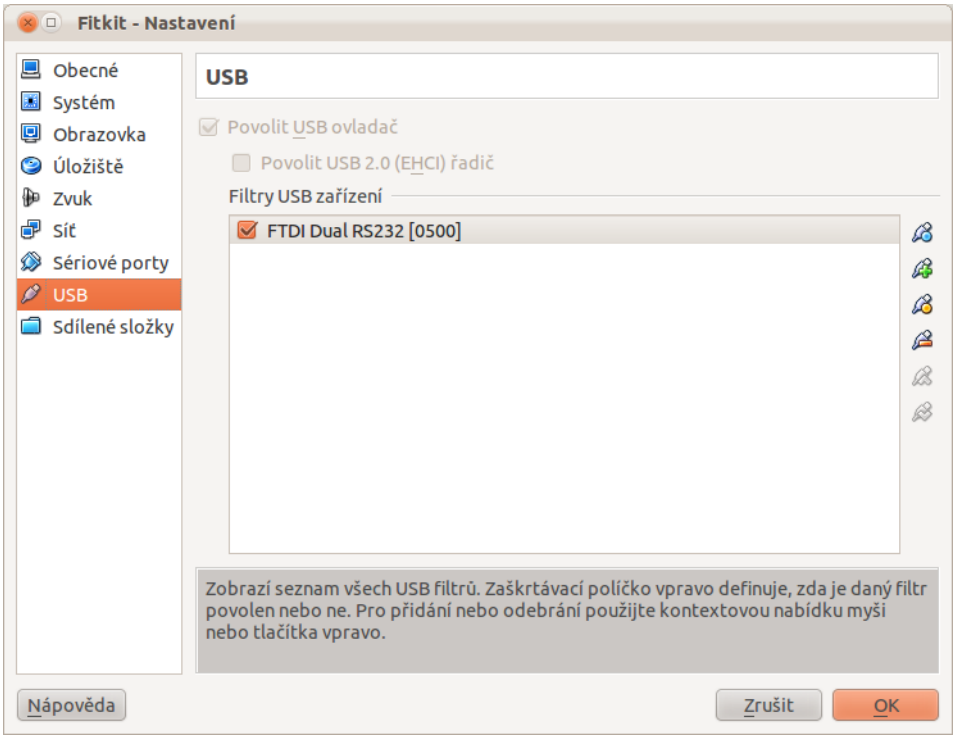
- MSPGCC – kompilátor zdrojových souboru pro mikrokontrolér MSP430
- Xilinx ISE – vývojové prostředí pro syntézu obvodů v jazyce VHDL do čipu FPGA
- ModelSIM – simulátor obvodů v jazyce VHDL
- QDevKit – prostředí pro práci s přípravkem FITkit
- Catapult C – vývojové prostředí pro syntézu algoritmů z jazyka C++ do jazyka VHDL
- Microsoft Visual C++ 2008 – kompilátor zdrojových souborů v jazyce C++

S ohledem na náročnost instalace všech těchto nástrojů a skutečnost, že ne všechny instalační soubory jsou volně ke stažení, byl pro Vás připraven obraz virtuálního stroje, kde jsou všechny tyto nástroje již připraveny. Obraz tohoto virtuálního stroje si můžete stáhnout z privátních stránek FITkitu. Pro spuštění tohoto stroje využijte volně dostupného programu VirtualBox. Pro pohodlnou práci je vhodné tomuto stroji vyčlenit alespoň 2GB paměti RAM. Pokud na svém počítači nemáte k dispozici dostatečné množství paměti, je možné využít počítače v CVT (doporučuji učebny N103 - N105).

Spuštění aplikace na přípravku FITkit

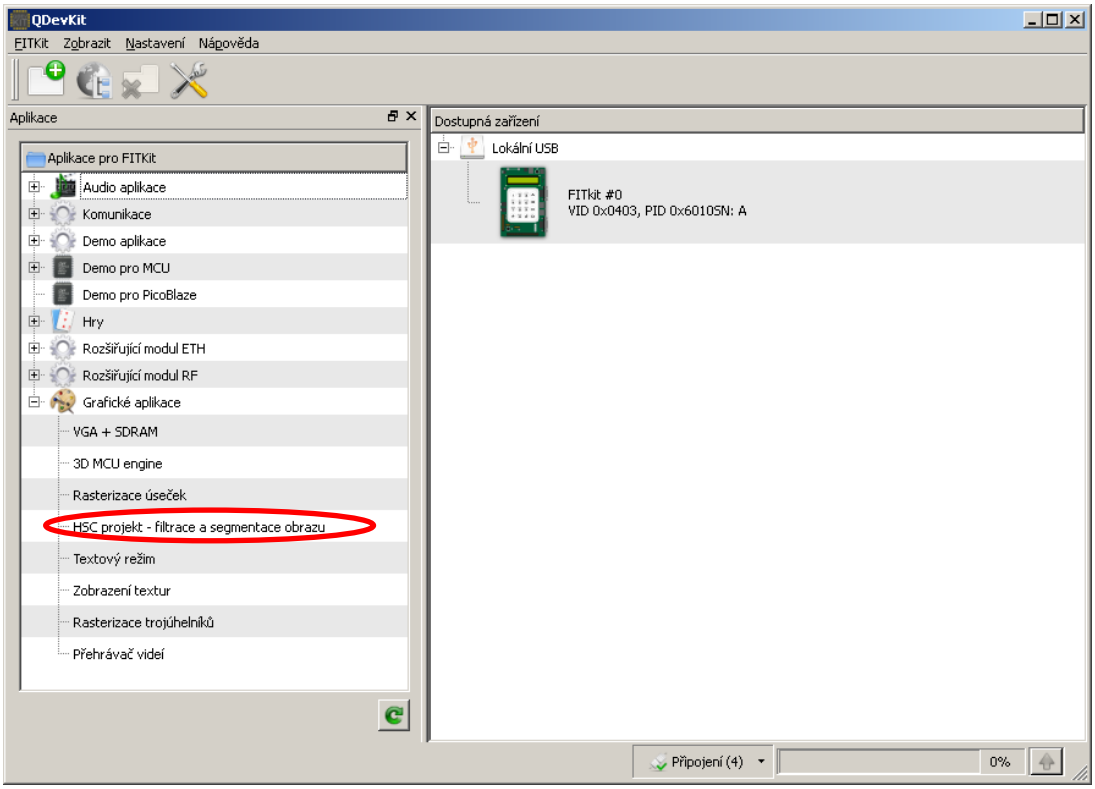
Postup pro spuštění aplikace na přípravku FITkit je následující:

1. Připojte FITkit k počítači.
2. Spusťte program VirtualBox a připojte obraz virtuálního stroje. V nastavení USB zařízení povolte FTDI Dual RS232 (viz obrázek) a virtuální stroj spusťte.



Poznámka: U některých typů operačních a verzí programu VirtualBox můžete narazit na problémy s připojením USB zařízení. Pokud se Vám to stane, napište prosím do diskusního fóra. Je možné, že některý z vašich spolužáků tento problém již řešil a nalezl vhodný postup řešení. Případně můžeme v diskusním fóru uvádět verze operačních systémů, kde se tento problém nevyskytuje.

3. Do adresáře *C:\FitkitSVN\apps\vga* nakopírujte rozbalený obsah archivu *projekt.zip*.
4. Spusťte si nástroj QDevKit. V záložce aplikace by se Vám měla objevit nová položka s názvem *HSC projekt – filtrace a segmentace obrazu* (viz obrázek).
5. Nyní můžete aplikaci přeložit, simulovat, naprogramovat do přípravku FITkit a následně spustit.



Poznámka k výpisu textu na terminál FITkitu

Pro výpis textu na terminál FITkitu ze strany mikrokontroléru se používají funkce *term_send_str()*, *term_send_num()*, apod. Někdy se vám může stát, že vypisovaný text nevypadá dle vašich představ. Například se vytiskne zalomení řádku navíc nebo se dokonce na konci zpracování vypíše obsah nápovědy. Jedná se o chyby, které prozatím nejsou v aplikaci QDevKit odladěny a na jejich odstranění se pracuje. Jako alternativu můžete použít ovládání FITkitu z příkazové řádky skrze příkazy (spuštěné v adresáři, kde je umístěn soubor *project.xml*):

`gmake load` – zajišťuje naprogramování přípravku FITkit

`gmake term` – zajišťuje otevření terminálu a spuštění aplikace

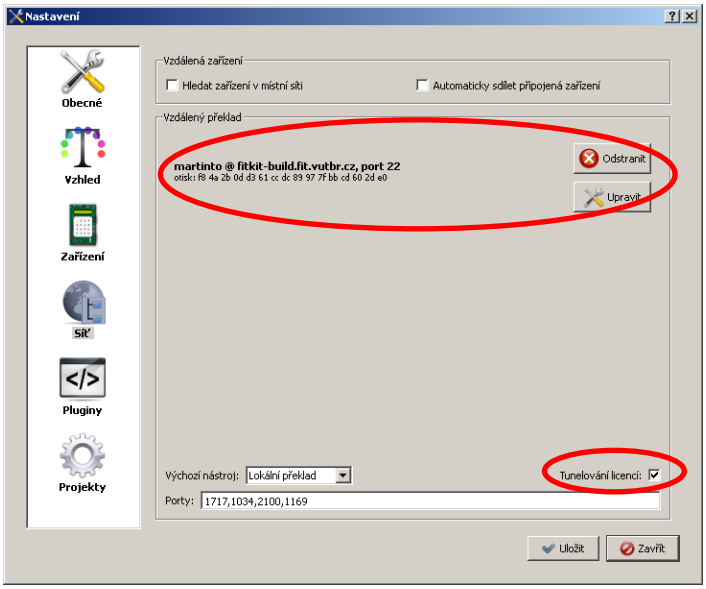
Při spuštění těchto příkazů je potřeba mít program QDevKit uzavřen (v opačném případě dochází k soupeření o přípravek a možným chybám).

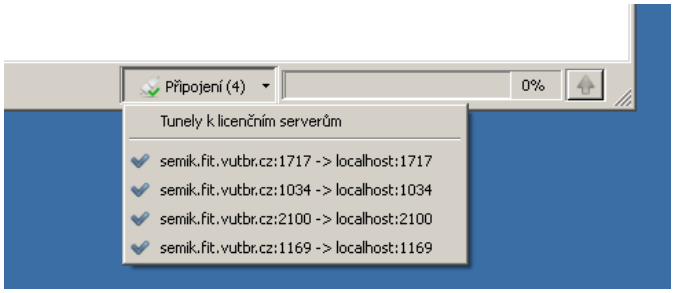
Syntéza souborů s využitím nástroje Catapult C

Automatický překlad souborů v prostředí Catapult C s využitím skriptu *directives.tcl*

Pro syntézu souborů z jazyka C do jazyka VHDL pomocí nástroje Catapult C postupujte dle následujícího návodu:

1. Nejprve si spusťte nástroj QDevKit. V menu *Nastavení → Síť* si nakonfigurujte vzdálený předklad a aktivujte volbu *Tunelování licencí* (viz obrázek). Po stisknutí tlačítka *Uložit* si QDevKit vytvoří ssh tunely směrem k licenčnímu serveru (stroj *semik.fit.vutbr.cz*) skrze stroj *fitkit-build.fit.vutbr.cz* a umožní tak nástrojům jako je ModelSIM, Precision, Catapult C a Xilinx ISE ověřit platnost licencí. Bez tohoto kroku skončí pokus o spuštění těchto programů neúspěšně. Stav navázání tunelů směrem k licenčnímu serveru si můžete ověřit ve spodní liště nástroje QDevKit (viz obrázek). Navázaní tunelovaného spojení skrze stroj *fitkit-build.fit.vutbr.cz* se nemusí vždy podařit, zejména pokud se nacházíte mimo síť VUT. V těchto případech zkuste upravit adresu serveru na *merlin.fit.vutbr.cz* nebo *eva.fit.vutbr.cz*.





2. Spusťte si nástroj Catapult C. V *Task Baru* zvolte volbu *Set Working Directory* a nastavte si vhodné umístění pro pracovní adresář, kde se vám budou ukládat všechny výstupní soubory překladu (doporučuji zvolit např. adresář *C:\FitkitSVN\apps\vga\segmentace\fpga*).
3. V hlavním menu zvolte možnost *File → Run Script...* a v příslušném adresáři se zdrojovými kódy (např. *src_filter/*) spusťte předpřipravený skript s názvem *directives.tcl*, který zajistí kompletní překlad zdrojových souborů a vytvoří výstupní VHDL soubor s názvem *mapped.vhdl*. Přesné umístění tohoto výstupního souboru vám program vypíše na posledních řádcích okna transkriptu. Měli byste vidět hlášení typu:
   -- Writing mapped netlist for 'filter' to file 'C:/FitkitSVN/apps/vga/segmentace/fpga/Catapult/filter.v1/mapped.vhdl'
   Tento výsledný soubor *mapped.vhdl* si zkopírujte do adresáře k původním zdrojovým kódům (např. *src_filter/*).
4. Před tím, než bude možné výstupní soubor *mapped.vhdl* použít pro syntézu, je potřeba jej upravit. Jedná se o odstranění řádků, které používají simulační knihovnu *simprim* (Cataput C tyto řádky automaticky vkládá pro účely simulace, pro syntézu však jejich přítomnost způsobuje chybu). Pro tyto účely je pro Vás připraven skript *convert.bat*, který zajistí potřebné změny. Tento skript postačí zkopírovat do adresáře, kde je uložen soubor *mapped.vhdl* a spustit jej.
5. Nyní lze v prostředí QDevKit spustit překlad aplikace a navazující fáze pro ověření aplikace na FITkitu.

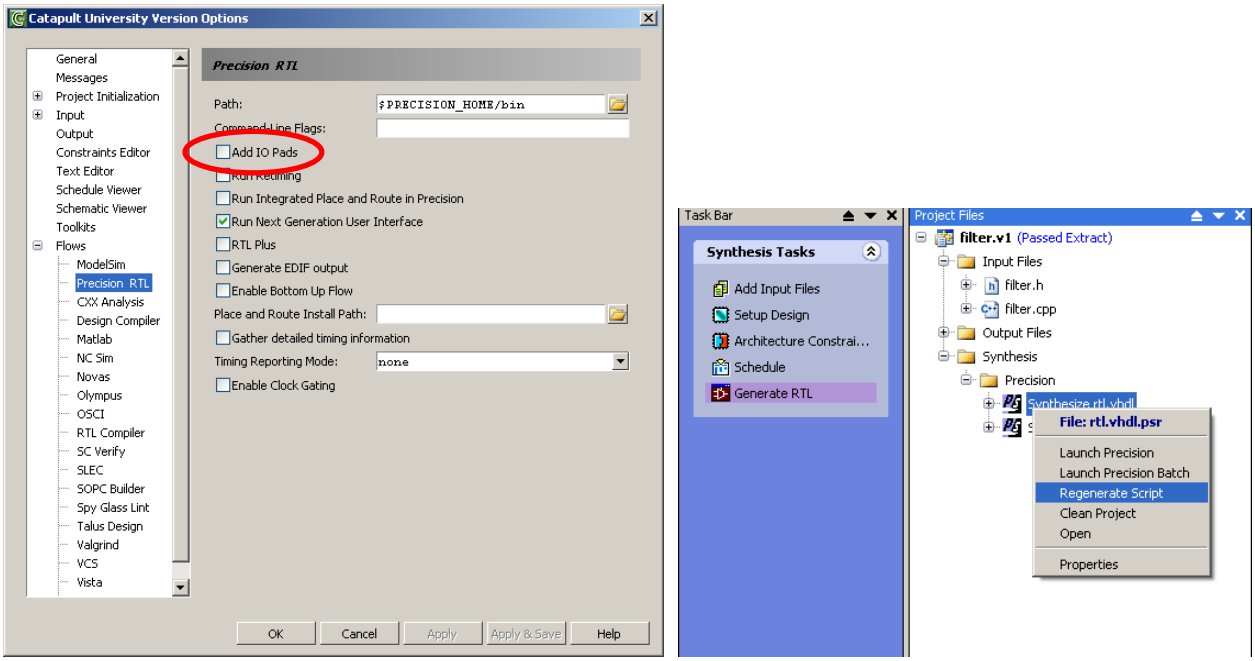
Manuální překlad souborů v prostředí Catapult C bez využití skriptu *directives.tcl*

Pokud z jakýchkoliv důvodů nebudete pro překlad moci využít předpřipravený skript *directives.tcl* a budete překlad provádět ručně, potom je potřeba abyste vzali v úvahu následující skutečnost. Výchozí nastavení nástroje Catapult C předpokládá, že výstupem syntézy bude komponenta, která se nahraje do čipu samostatně, bez jakýchkoliv dalších komponent. Ve výstupním VHDL souboru *mapped.vhdl* jsou proto připojeny elementy typu IBUF a OBUF, které

zajišťují připojení komponenty přímo k pinům FPGA čipu. Toto výchozí chování je pro použití v tohoto projektu nevhodné, neboť výstupní VHDL soubory se připojují ještě k dalším komponentám v systému.

Pro získání správných výsledků je potřeba zakázat nástroji vkládání I/O Pad elementů do výstupního souboru dle následujícího postupu:

1. Skrze hlavní menu spusťte dialogové okno *Tools→ Set Options→ Flows→ Precision RTL* a v tomto okně odstraňte volbu *Add IO Pads* (viz obrázek).
2. Jakmile při vývoji dospějete do fáze, kdy již máte vygenerováno RTL (*Task Bar → Generate RTL*), klikněte pravým tlačítkem myši na položku *Projekt Files → Synthesis → Precision → Synthesize.rtl.vhdl* a zvolte volbu *Regenerate script* (viz obrázek).
3. Jako poslední krok zvolte ve stejném menu položku *Lanch Precision Batch*. Po dokončení procesu překladu získáte požadovaný soubor *mapped.vhdl*.



Pokud pro překlad používáte vlastní skript *directives.tcl* a neradi opakovaně klikáte v menu, potom můžete stejného efektu dosáhnout přidáním následujících tří řádků za příkaz *go extract*:

```
options set Flows/Precision addio false
flow run /Precision/generate
flow run /Precision/precision -shell -file ./rtl.vhdl.psr -run_state mapped
```

Simulace a verifikace komponent

Pro jednotlivé komponenty systému jsou pro vás v projektu předpřipraveny testbench soubory, které ověřují jejich základní funkcionalitu. Současně s testbench soubory v adresářích také naleznete soubory *directives_sim.tcl*, které slouží pro překlad komponenty a její následnou simulaci. Tento skript kromě přípravy souborů a jejich překladu také aktivuje prostředí *SCVerify* včetně signálu *Transaction Done*.

Verifikace originálního kódu oproti modifikovanému kódu

Verifikaci spustíte výběrem volby *Project Files → Verification → MS Visual C++ 9.0 → Original Design + Testbench*. V průběhu testování vašeho modifikovaného kódu oproti originálnímu kódu, můžete pocítit potřebu vypisovat si určité ladící informace ze strany modifikovaného kódu. Obvykle toto nelze přímo realizovat, neboť modifikovaný kód je součástí kompilace výsledného obvodu a na úrovni obvodu nelze používat funkce typu *printf* nebo *cout*. Tento problém lze však elegantně vyřešit skrze podmíněný překlad a direktivu kompilátoru *#ifdef* (viz následující příklad):

```
#ifdef CCS_DUT_SYSC
   cout << "Zprava pouze pro ucely verifikace" << endl;
#endif
```

Verifikace modifikovaného kódu oproti RTL schématu komponenty

Verifikaci spustíte výběrem volby *Project Files → Verification → ModelSim → RTL VHDL output 'rtl.vhdl' vs Untimed C++*. Verifikace modifikovaného kódu oproti RTL schématu je realizována skrze prostředí ModelSim. V rámci ModelSimu probíhá i kompilace původního kódu v C/C++ a některé konstrukce jsou pro něj těžce stravitelné, např. podmíněné přirazení hodnot do proměnné:

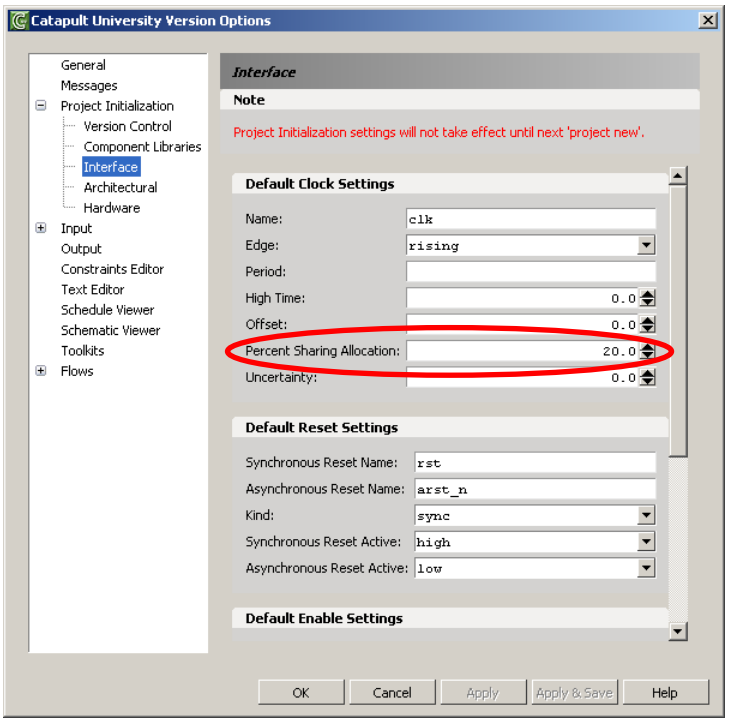
```
ac_int<4,false> promenna;
promenna = (podminka) ? hodnota1 : hodnota2;
```

překlad bude fungovat pouze, pokud hodnoty explicitně přetypujete:

```
promenna = (podminka) ? (ac_int<4,false>)hodnota1 : (ac_int<4,false>)hodnota2;
```

Neúspěšný překlad aplikace v prostředí QDevKit

V některých případech se vám v prostředí QDevKit nemusí podařit aplikaci úspěšně přeložit. Pokud v terminálu FITkitu objevíte hlášení typu „*WARNING: design did not meet timing*“, potom to znamená, že se nepodařilo rozmístit jednotlivé elementy obvodu tak, aby dodrželi frekvenci 25MHz. Jedním z důvodů může být také to, že nástroj Catapult C neodhadl správně zpoždění jednotlivých operací v CDF grafu (principiálně to ani přesně udělat nelze) a naplánoval jich do jednoho C-stepu příliš mnoho. Vyřešit tento problém lze například tak, že v nastavení Catapultu zvýšíme procento režie pro C-step. Zvýšení této hodnoty lze provést (před založením nového projektu) skrze hlavní menu *Tools → Set Options → Project Initialization → Default Clock Setting* a volbu *Percent Sharing Alocations* (viz obrázek). Výchozí nastavení je 20%, což nemusí být pro FPGA čipy dostačující.



Pokud pro překlad používáte vlastní skript *directives.tcl* a neradi opakovaně klikáte v menu, potom můžete stejného efektu dosáhnout přidáním následujícího řádku za příkaz *new project*:

```
directive set -CLOCK_OVERHEAD 20.000000
```

Pokyny pro vypracování technické zprávy

Technická zpráva musí obsahovat následující části:

1. ***Hlavička*** obsahující ***název předmětu***, ***školní rok***, ***jméno***, ***příjmení*** a ***login***.
2. ***Tabulka a graf výsledků analýzy algoritmu z programu gprof.*** V tabulce pro jednotlivé funkce uveďte, kolik procent času strávil procesor ve funkci z celkové doby výpočtu. Hodnoty z tabulky vyneste do sloupcového grafu, kde každá funkce bude reprezentována jedním sloupcem.
3. ***Tabulka ukazující rozdělení aplikace mezi hardware a software.*** Vytvořte dva sloupce, kde do prvního vypište seznam funkcí původního algoritmu, které jste ponechali v mikrokontroléru. Do druhého pak uveďte funkce, které jste přesunuli do FPGA.
4. ***Tabulka popisující využití adresového prostoru sdílené paměti.*** V této tabulce naznačte, které položky sdílené paměti jste využili a stručně popište, pro jaký účel byly použity. U každé položky také uveďte, zda se z pohledu mikrokontroléru chová jako vstup (pouze pro čtení), výstup (pouze pro zápis), nebo obojí vstup/výstup (čtení i zápis).
5. ***Tabulka shrnující vlastnosti obvodu uvnitř FPGA.***
    a. Uveďte vlastnosti vaší komponenty pro filtraci a segmentaci obrazu (soubor *fpga/src_filter/filter.cpp*), zejména inicializační interval hlavní smyčky a latenci obvodu (lze odečíst z Ganttova diagramu).
    b. Na základě výsledků syntézy uveďte množství spotřebovaných zdrojů FPGA čipu v počtech *Flip Flops*, *LUTs* a *Slices*. Tyto informace najdete v souboru *build/fpga/med_filtr.map.mrp*.
6. ***Tabulka porovnávající vlastnosti čistě softwarové implementace na mikrokontroléru MSP430 a implementace rozdělené mezi hardware a software.*** V této tabulce uveďte pro obě realizace:
    a. Průměrnou dobu pro zpracování jednoho pixelu. Pozn.: U hardwarového řešení lze tuto hodnotu odvodit přesně z hodnoty latence obvodu (viz Catapult) a frekvence 25MHz.
    b. Počet bodů zpracovaných za vteřinu. Pozn.: U hardwarového řešení lze tuto hodnotu odvodit z hodnoty propustnosti obvodu (viz Catapult) a frekvence 25 MHz.
    c. Hodnotu zrychlení, která vyjadřuje kolikrát je řešení využívající hardware rychlejší oproti čistě softwarovému řešení. Ve sloupci u softwarového řešení uveďte hodnotu 1.
7. ***Shrnutí*** – zhruba pětiřádkový komentář k dosaženým výsledkům. Při hodnocení systému berte v úvahu nejen rychlost zpracování, ale i potřebnou velikost čipu, cenu nebo předpokládanou spotřebu výsledného řešení. Zamyslete se, co je úzkým místem navrženého systému, a jakým způsobem by bylo možné zpracování ještě více urychlit, případně za jakou cenu. Pokud se vám nepodaří zadání zcela splnit, uveďte prosím, kterých bodů se to týkalo.

Celá technická zpráva by svým rozsahem neměla překročit 2 strany formátu A4.