Technická zpráva

Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Aplikační příručka k vývojové desce Uni1P a modulům DX64

Ing. Jan Kloub
kloub@utia.cas.cz, +420-2-6605 25022502

Prostředky pro rychlý vývoj
HW-akcelerovaných vestavěných aplikací zpracování obrazu a videa
1ET400750408

Department of Signal Processing
http://zs.utia.cz

Obsah

1 Úvod **1**

2 Stručný popis architektury Uni1P **1**

3 Modul DX64 **1**
3.1 Boot DSP na modulu DX64 2
3.2 Paměť SDRAM 4

4 Koncept paměťových portů implementovaných v FPGA připojeném k DSP **4**
4.1 Registry paměťových portů 5
4.2 Vstupní a výstupní paměťové porty výpočetního jádra FPGA 6

5 Komunikace DSP s FPGA **6**
5.1 Podpora přenosů dat v jazyce C 8

6 Komunikace hostitelského systému s deskou Uni1P a moduly DX64 **9**
6.1 Komunikace s modulem DX64 přes HPI 9
6.2 Přístup ke konfiguračnímu rozhraní obvodů FPGA 10

7 Implementace hranového detektoru **10**
7.1 Návrh hranového detektoru pomocí nástroje Synplify DSP 10
7.2 Propojení hranového detektoru a paměťových portů 11
7.3 Využití částečné dynamické rekonfigurace 11
7.4 Program pro DSP 12
7.5 Program na straně PC 13

7.6 Výsledky 14

8 Poděkování 15

9 Výpis CDROM 18

A Generování bitstreamu 20

B Nahrání bitstremů do jednotlivých FPGA a programu pro DSP 21
B.1 Konfigurace Systémového FPGA 21
B.2 Konfigurace obvodů FPGA 21
B.3 Dynamická rekonfigurace na Uni1P 22
B.4 Generování binárního souboru programu pro DSP a jeho zavedení 23

C Výběr zdroje hodinového signálu pro FPGA na DX64 23

D Funkce knihovny libUni1P 24

E Aplikační poznámky 26

Revize

Revize	Datum	Autor	Popis změn v dokumentu
0	6.11.2008	Kloub	Vytvoření dokumentu
1			
2			
3			
4			

1 Úvod

Tento dokument popisuje architekturu karty Uni1P a akceleračních modulů DX64 a způsob jejich využití. Je popsán způsob návrhu aplikací pro platformu využívající kartu Uni1P a její moduly. Součástí dokumentu je popis implementované aplikace pro hranovou detekci obrazu.

Struktura dokumentu je následující: kapitola 2 obsahuje stručný popis architektury karty Uni1P, kapitola 3 popisuje architekturu modulu DX64, kapitola 4 popisuje koncept použití paměťových portů pro komunikaci mezi DSP a FPGA na modulech DX64, kapitola 5 popisuje komunikaci programu DSP s paměťovými porty v FPGA, kapitola 6 popisuje komunikaci mezi hostitelským systémem a deskou Uni1P a jejími moduly a kapitola 7 popisuje implementaci hranového detektoru.

2 Stručný popis architektury Uni1P

Uni1P je vývojová karta určená pro připojení na sběrnici PCI osobního počítače. Rozhraní PCI je implementováno pomocí obvodu PLX9054, který zajišťuje komunikaci mezi jednotlivými komponentami karty a osobním počítačem.

Softwarová podpora PCI rozhraní desky je v operačním systému Windows zajištěna pomocí ”WinDriver PCI for Windows Driver Development Toolkit” [1].

Uni1P obsahuje tři obvody FPGA, jeden obvod CPLD a čtyři sloty pro moduly DX64 (viz Obr. 1). Funkce jednotlivých obvodů je následující:

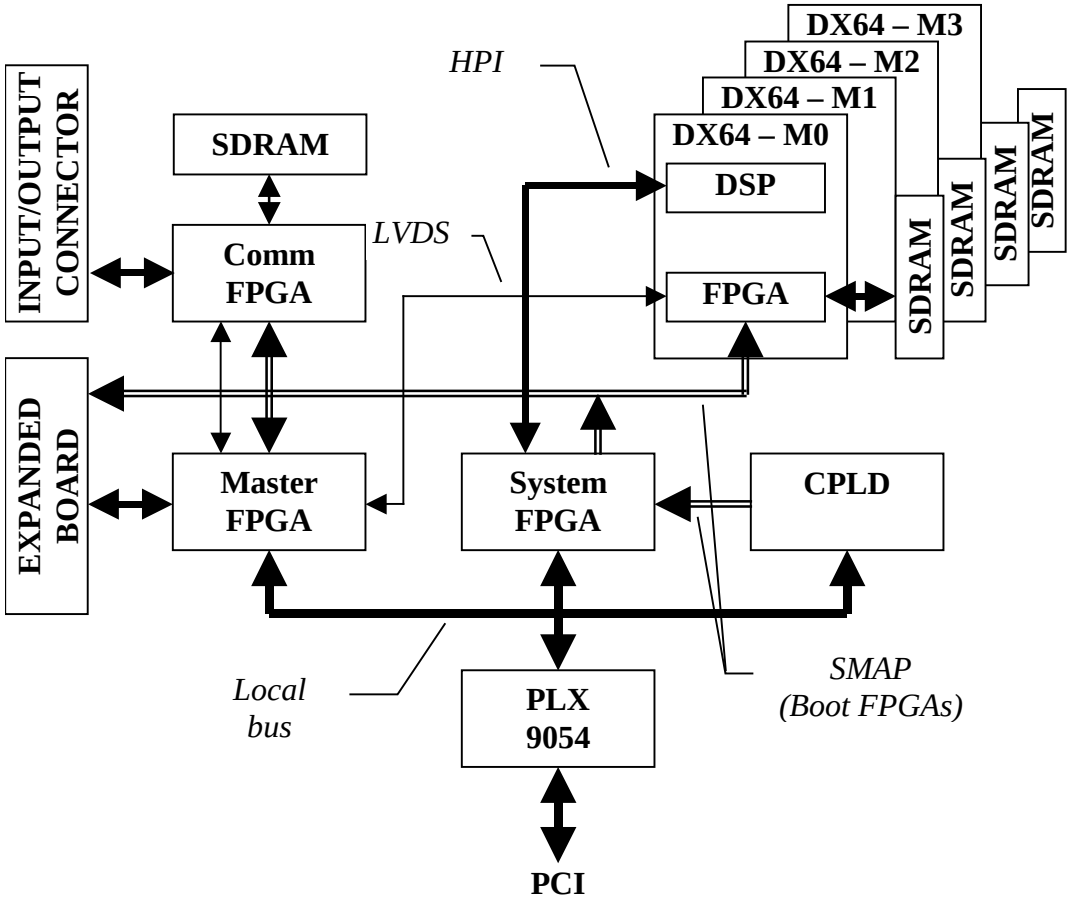
- CPLD slouží k řízení konfiguračního rozhraní systémového FPGA (System FPGA) v režimu ”Slave SelectMAP”.
- ”System FPGA” řídí konfigurační rozhraní zbývajících FPGA na desce Uni1P a obvodů FPGA na modulech DX64. Systémové FPGA používá pro programování ostatních obvodů FPGA režim ”Slave SelectMAP” (viz obrázek 2).
- ”Master FPGA” je použito pro komunikaci s moduly DX64 pomocí LVDS a komunikaci s externím rozhraním.
- ”Comm FPGA” je určeno pro obsluhu paměťového rozhraní SDRAM paměti a obsluhu rozšiřujícího konektoru pro obecné užití.
- Modul DX64 obsahuje signálový procesor (DSP) firmy Texas Instruments TMS320C6416, obvod FPGA a paměť SDRAM. K DSP je připojena paměť SDRAM, obvod FPGA (EMIFB rozhraní) a deska Uni1P (HPI rozhraní). DSP je pomocí HPI propojen se systémovým FPGA.

3 Modul DX64

Modul DX64 je osazen signálovým procesorem firmy Texas Instruments TMS320C6416 (DSP), obvodem FPGA firmy Xilinx Virtex-II (xc2v250-4fg256 nebo xc2v1000-4fg256) a pamětí SDRAM (2 x SDRAM_8M32).

Obvod FPGA a DSP jsou spolu propojeny pomocí rozhraní EMIFB a zároveň jsou propojeny přes konektor modulu s deskou Uni1P. DSP je propojeno s Uni1P pomocí rozhraní HPI. Obvod FPGA je propojen s Uni1P pomocí následujících rozhraní:

- Rozhraní podporující standard LVDS.
- Rozhraní propojující sousední moduly.
- Rozhraní pro paměť SDRAM osazené na Uni1P.



Blokové schéma Uni1P

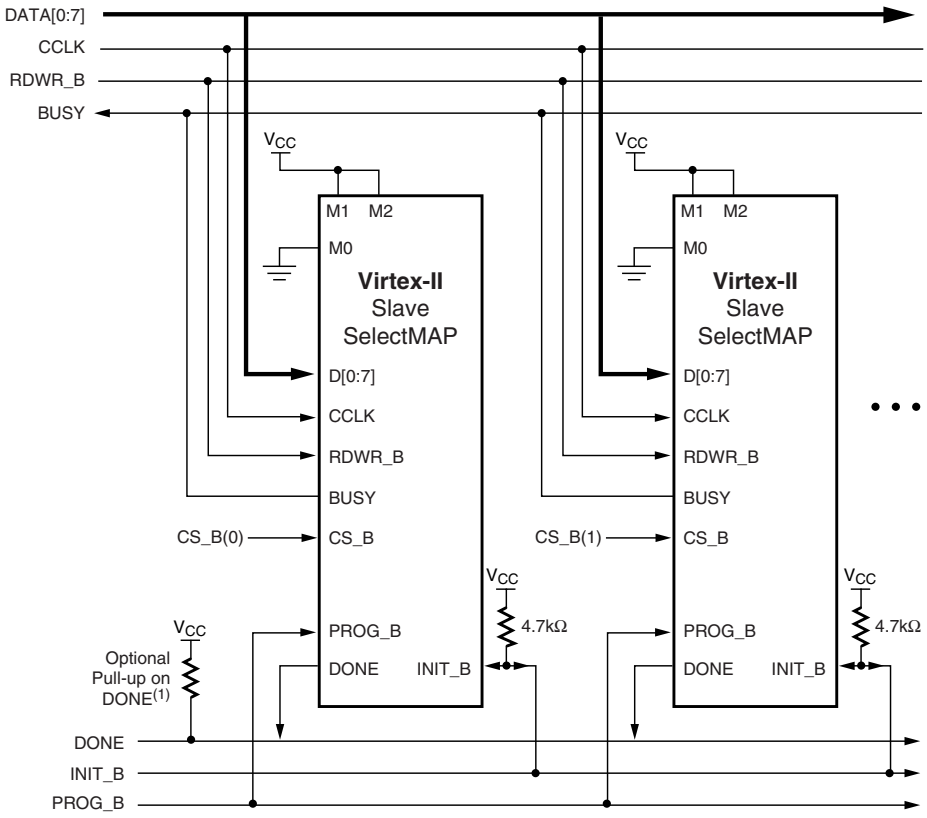
Obrázek 1: Architektura vývojové desky Uni1P

3.1 Boot DSP na modulu DX64

DSP je nakonfigurováno tak, že bootuje z externí paměti programu ROM připojené na rozhraní EMIFB CE1. Protože k rozhraní EMIFB je připojen obvod FPGA, musí zastoupit funkci programové ROM. Mimo funkce programové ROM zastupuje FPGA i řízení signálu pro reset a nemaskovatelné přerušení DSP a výběr násobiče hodinové frekvence DSP (viz Příloha C).

Pro spuštění programu na DSP je nutné provést následující kroky:

1. Pomocí aplikace `uni1p.exe` je nutné nahrát příslušný bitstream do systémového FPGA desky Uni1P. Systémové FPGA zajišťuje komunikaci s DSP na jednotlivých modulech pomocí HPI a řídí konfigurační rozhraní obvodů FPGA osazených na modulech DX64 a desce Uni1P (viz Příloha B).
2. Vygenerovat bitstream pro FPGA na modulu DX64 s příslušnými parametry aplikace `bitgen` (viz. Příloha A).
3. Bitstream nahrajeme pomocí aplikace `uni1p_fpga.exe` (viz Příloha B).
4. DSP je držen v resetu, dokud není nahrán správný bitstream do FPGA osazeném na modulu. Pokud je třeba zresetovat DSP za běhu lze tak provést zápisem hodnoty 0x1 na adresu 0x60000000 DSP, která odpovídá adrese EMIFB CE0 rozhraní. Zápis se provede pomocí HPI rozhraní. Logika implementovaná v FPGA začne generovat resetovací signál. Hodinový signál pro FPGA je generován výstupem BECLKOUT obvodu DSP.



Obrázek 2: Způsob propojeni konfiguračního rozhraní obvodů FPGA v režimu ”Slave SelectMAP” (převzato z [2]

Pokud je DSP v resetovacím stavu hodinový signál na BECLKOUT není generován. Pro zachování funkce synchronního návrhu v FPGA je zvolen hodinový signál z externího oscilátoru modulu (viz Příloha C).

5. DSP načte program (1 KB) z rozhraní EMIFB CE1 a uloží si jej do vnitřní paměti na adrese 0 a poté se z adresy 0 spustí vykonávání instrukcí programu.
6. Program vykoná následující kroky:
   (a) Nastaví EMIFB CE2 rozhraní do režimu SSRAM.
   (b) Povolí NMI v ”Interrupt Enable Register”.
   (c) Uvede DSP do IDLE stavu.
7. Vygenerovat binární soubor s novým programem pro DSP (viz Příloha B.4).
8. Binární soubor nahrajeme pomocí `uni1p_dsp.exe` od adresy 0 (viz Příloha B.4).
9. Vygenerujeme nemaskovatelné přerušení (NMI) pomocí obdobného postupu jako v bodu 4 - na adresu 0x60000000 nahrajeme hodnotu 0x2 přes HPI.
10. DSP začne vykonávat program od adresy 0, protože obslužná rutina NMI v novém programu obsahuje skok na adresu 0.

3.2 Paměť SDRAM

Na modulu DX64 je osazena paměť SDRAM o velikosti 2 x 1MB. Paměť je připojena k DSP pomocí 64 bitového rozhraní EMIFA, a proto je její adresní prostor mapován od adresy 0x80000000h ([6]). Pro správnou funkci paměti je nutné nastavit konfigurační registry rozhraní EMIFA. Hodnoty konfiguračních registrů pro správnou funkci paměti jsou uvedeny v tabulce 1.

Registr	Hodnota registru
GBLCTL	0x000000a0
CE0CTL	0x000000d0
SDTIM	0x00000300
SDEXT	0x00054528
SDCTL	0x53116000

Tabulka 1: Hodnoty konfiguračních registrů rozhraní EMIFA pro správnou funkci paměti SDRAM

4 Koncept paměťových portů implementovaných v FPGA připojeném k DSP

Mezi DSP a FPGA dochází k blokovým datovým přenosům přes rozhraní EMIFB CE2. Pro datové přenosy jsou v FPGA implementovány paměťové porty, které obsahují dvouportové paměti a několik řídících registrů (záleží na implementaci). Jeden port paměti je připojen k EMIFB a druhý port je připojen ke komponentě zpracovávající data v FPGA.

Paměti jsou logicky rozděleny do sektorů o velikosti 512 B. Ke každé paměti náleží adresový registr, registr určující platnost dat a případně další registry v závislosti na implementaci aplikace. Registry a rozhraní paměti je mapováno do adresního prostoru EMIFB. Popis a význam registrů je popsán v kapitole 4.1.

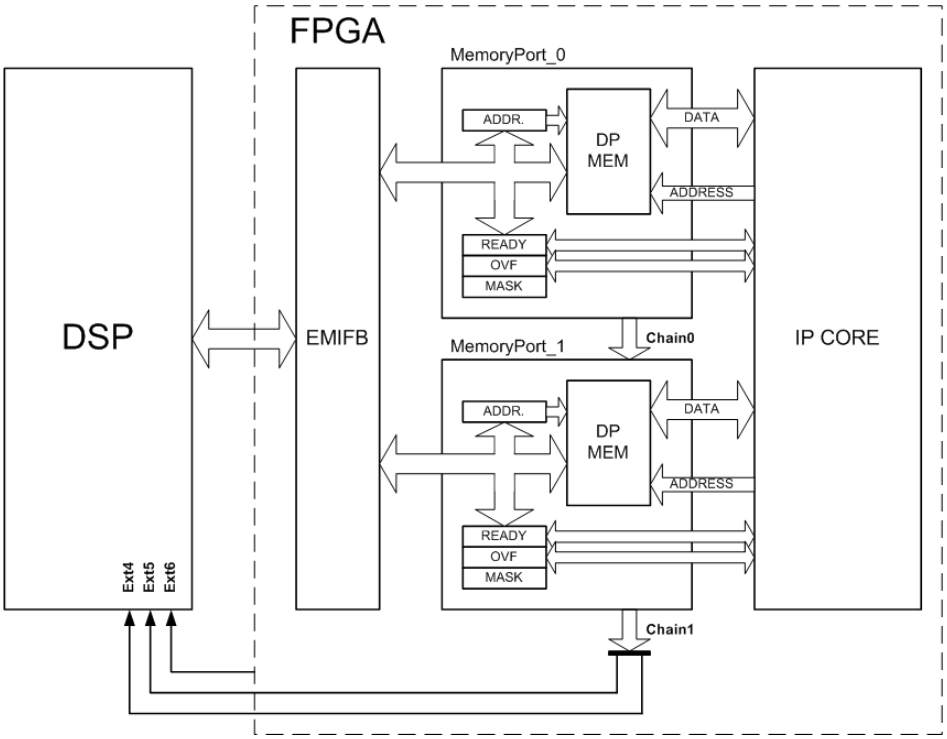
Datový přenos mezi DSP a FPGA lze rozdělit (nezávisle na směru přenosu) do následujících kroků:

- Zápis hodnoty počáteční adresy pro čtení/zápis do adresového registru paměťového portu.
- Zápis/čtení dat z rozhraní paměti (při každém přístupu k paměťovému rozhraní je inkrementován adresový registr).
- Zápis hodnoty do registru určující platnost dat (bitová mapa určující platnost jednotlivých datových sektorů).

Pro efektivní realizaci přenosů dat z paměťových portů je využit EDMA řadič DSP. Přenos je inicializován na základě hodnot přerušovacích signálů generovaných v FPGA. Pro přerušení DSP jsou využívány tři signály pro externí přerušení procesoru EXT4, EXT5 a EXT6. Signál EXT4 je využit k indikaci platných bloků dat paměťových portů, EXT5 je využit k indikaci paměťových bloků v kterých došlo k přepisu platných dat a EXT6 je využit pro ostatní účely závislé na konkrétní aplikaci. Přerušení pro indikaci platných bloků dat lze maskovat pomocí maskovacího registru.

Paměťové porty jsou navzájem propojeny do řetězce (”chain”). Pomocí logického součtu propagují signály určující zda port obsahuje platná nebo přepsaná data až k poslednímu v řetězci. Výstup řetězce posledního paměťového portu je připojen k signálům EXT4 a EXT5.

Na obrázku 3 je znázorněn základní koncept s jedním vstupním a jedním výstupním portem.



Obrázek 3: Základní koncept paměťových portů pro blokový přenos dat

4.1 Registry paměťových portů

Paměťové porty obsahují několik registrů mapovaných do adresního prostoru EMIFB připojeného DSP. Základními registry jsou datový (MEMPORT_DATA) a adresní (MEMPORT_ADDR) registr, které umožňují přístup do blokových pamětí paměťových portů. Dalšími registry jsou MEMPORT_READY, MEMPORT_MASK a MEMPORT_OVF, které slouží k udržení informací o platnosti dat v jednotlivých paměťových sektorech. Význam jednotlivých registrů je shrnut v tabulce 2.

Název registru	Význam registru
MEMPORT_ADDR	Registr obsahující aktuální adresu do paměti portu. Adresa je inkrementována s každým přístupem do paměti automaticky.
MEMPORT_DATA	Tento registr je mapován přímo na datový vstup nebo výstup paměti portu v závislosti na tom, zda se z registru čte či zapisuje.
MEMPORT_READY	Registr obsahuje bitovou mapu, která určuje jaký sektor v paměti obsahuje platná data.
MEMPORT_MASK	Registr je využíván k maskování bitů registru MEMPORT_READY (není použit pro maskování registru MEMPORT_OVF) pro generování signálu určující platnost dat předávaného v řetězci portů.
MEMPORT_OVF	Registr obsahuje bitovou mapu, která určuje jaký sektor v paměti byl přepsán. Na základě obsahu registru je generován signál určující zda došlo k přepisu dat a ten je předáván v řetězci portů.

Tabulka 2: Význam registrů paměťových portů

Paměťové porty jsou umisťovány do vlastního adresního prostoru, kde je každý port určen počátečním indexem svých sektorů a jejich počtem. Každý port obsahuje všechny výše zmíněné registry (viz tabulka 2). Pokud je hodnota registru MEMPORT_ADDR daného portu mimo rozsah svého adresního prostoru,

nelze přistupovat k pamětem portu a registr MEMPORT_DATA je nastaven pro případ jeho čtení na nulovou hodnotu.

Všechny registry paměťových portů jsou mapovány na stejné pozice v paměťovém prostoru EMIFB. Aby bylo možné číst data z adresního prostoru portů jsou všechny datové výstupy portů sečteny logickým součtem, protože neadresované porty obsahují v registru MEMPORT_DATA nulovou hodnotu a naadresovaný port vybaví data z paměti.

Adresy registrů v paměťovém prostoru rozhraní EMIFB jsou definovány pomocí konstant ve zdrojových kódech ve VHDL a jsou zadávány jako pozice slov (2 bajty) od počátku paměťového prostoru rozhraní EMIFB (0x68000000h). Například adresa registru MEMPORT_ADDR je definována následovně:

```
constant MEMPORT_ADDR_LOW_ADDR_C        : natural := 16;--0x10h
constant MEMPORT_ADDR_HIGH_ADDR_C       : natural := 17;
```

Odpovídající adresa registru MEMPORT_ADDR v rámci celého paměťového prostoru DSP je 0x68000020h (0x68000000h + (0x10h * 2)).

Překryv registrů obsahující bitové mapy je realizován obdobně a to tak, že příslušný port zapisuje do bitové mapy od pozice dané svým indexem do pozice dené svým indexem a počtem sektorů. Na ostatní pozice mapy jsou zapsány nuly. Logickým součtem tak získáme přehled o všech sektorech přes celý paměťový prostor portů v jednom datovém slově.

4.2 Vstupní a výstupní paměťové porty výpočetního jádra FPGA

Paměťové porty se dají rozdělit na vstupní a výstupní z hlediska připojení implementované funkce v FPGA. Vstupním portem uvažujeme takový port, do něhož jsou ukládány vstupní data pro funkci v FPGA pomocí DSP. V tomto případě DSP nastavuje registr MEMPORT_READY, tak aby implementovaná funkce zpracovala příslušná data. Příslušné pozice v bitové mapě jsou po zpracování dat nulovány ze strany funkce FPGA. Obdobně se zachází s registrem MEMPORT_OVF.

Výstupním portem se myslí takový port, do kterého funkce v FPGA zapisuje výstupní data a nastavuje příslušný registr MEMPORT_READY. Pokud jsou data ve výstupním portu platná, může je DSP vyčíst. Za pomocí registru MEMPORT_READY a MEMPORT_MASK může být zahájen přenos dat do DSP na základě externího přerušení EXT4 nebo EXT5, jak bylo popsáno výše. Způsob jakým DSP realizuje přenosy dat je popsán v kapitole 5

Implementace vstupních a výstupních portů se liší pouze ve způsobu práce s registry MEMPORT_READY a MEMPORT_OVF. Význam registru MEMPORT_READY je interpretován v této modifikaci jiným způsobem. Pro vstupní port obsahuje bitová mapa v registru MEMPORT_READY informace o tom, které sektory jsou připraveny pro zápis vstupních dat (označuje sektory, které neobsahují žádná platná data) a pro výstupní port obsahuje informaci o tom, které sektory obsahují platná data.

5 Komunikace DSP s FPGA

Signálový procesor komunikuje s výpočetním jádrem v FPGA pomocí paměťových portů, jak bylo popsáno v kapitole 4. K přenosu dat je využito nezávislého EDMA (Enhanced DMA) řadiče. Po přenosu dat pomocí EDMA je využit registr MEMPORT_READY k vzájemnému informování DSP a FPGA o platnosti dat.

Pro zápis dat do paměťových portů FPGA je užit následující scénář:

1. Určení počtu volných sektorů pro zápis vstupních dat pomocí registru MEMPORT_READY.
2. Určení počáteční adresy prvního volného paměťového sektoru.
3. Určení maximální délky kontinuálně přenášených dat. Maximální délka dat pro přenos je určena počtem za sebou jdoucích volných sektorů.

4. Povolení přerušení generovaného EDMA řadičem po dokončení přenosu.
5. Povolení generování přerušení pro zvolený kód přenosu EDMA řadiče (EDMA Complete Code). Každý přenos EDMA řadiče je identifikován svým kódem.
6. Vyplnění parametrů určující režim přenosu, číslo kódu přenosu, povolení generování přerušení po dokončení přenosu, prioritu přenosu, zdrojovou adresu, délku přenášených dat, cílovou adresu a velikost přenášeného elementu (bajt, slovo, dvojité slovo) atd.
7. Parametry jsou zapsány do příslušných registrů EDMA řadiče a jakmile je to možné je zahájen nezávislý přenos dat na vlastním jádře procesoru.
8. Po dokončení přenosu je vyvoláno přerušení.
9. Na základě přerušení je upravena hodnota registru MEMPORT_READY tak, aby bylo FPGA informováno o platnosti dat v sektorech, do nichž se zapisovalo.

Čtení dat z paměťových sektorů FPGA může být inicializováno buď sledováním stavu registru MEMPORT_READY nebo pomocí přerušení generovaného FPGA (EXT4 a EXT5).

Přerušení je generováno na základě hodnoty registrů MEMPORT_READY, MEMPORT_OVF a MEMPORT_MASK, jak bylo popsáno v kapitole 4. S využitím přerušení může DSP provádět jiné užitečné operace dokud nejsou zpracována data pomocí FPGA.

Pro čtení dat z paměťových portů FPGA na základě externího přerušení procesoru je užit následující scénář:

1. Povolení přerušení pro externí přerušovací signály DSP (EXT4, EXT5).
2. Povolení přerušení generovaného EDMA řadičem po dokončení přenosu.
3. Povolení generování přerušení pro zvolený kód přenosu EDMA řadiče (EDMA Complete Code). Každý přenos EDMA řadiče je identifikován svým kódem.
4. Nastavení registru MEMPORT_MASK dle potřeb konkrétní aplikace.
5. Pokud FPGA zpracuje data a zapíše je do paměťových sektorů, je upraven registr MEMPORT_READY nebo MEMPORT_OVF a je generováno přerušení.
6. Po vyvolání přerušení je spuštěna obslužná rutina.
7. Na základě hodnoty MEMPORT_READY nebo MEMPORT_OVF je určena adresa a délka přenášených dat.
8. Povolení přerušení generovaného EDMA řadičem po dokončení přenosu.
9. Povolení generování přerušení pro zvolený kód přenosu EDMA řadiče.
10. Vyplnění parametrů určující režim přenosu, číslo kódu přenosu, povolení generování přerušení po dokončení přenosu, prioritu přenosu, zdrojovou adresu, délku přenášených dat, cílovou adresu a velikost přenášeného elementu (bajt, slovo, dvojité slovo) atd.
11. Parametry jsou zapsány do příslušných registrů EDMA řadiče a jakmile je to možné je zahájen nezávislý přenos dat na vlastním jádře procesoru.
12. Po dokončení přenosu je vyvoláno přerušení.
13. Na základě přerušení je upravena hodnota registru MEMPORT_READY nebo MEMPORT_OVF tak, aby bylo FPGA informováno o uvolněných sektorech dat.

5.1 Podpora přenosů dat v jazyce C

Program signálového procesoru je psán v jazyce C. Pro využití všech vlastností procesoru je k dispozici knihovna CSL (Chip Support Library). S využitím CSL jsou inicializovány i přenosy pomocí EDMA řadiče.

Přenos dat pomocí EDMA

K inicializaci EDMA přenosů je využívána funkce `EDMA_qdmaConfigArgs();`, které jsou předány parametry přenosu. První parametr je hodnota, která bude zapsána do registru "Options Parameter" (OPT) [6] řadiče EDMA. Registr OPT obsahuje bitové pole určující prioritu přenosu, velikost přenášených slov, adresní mód a další. Následující příklad ukazuje volání funkce `EDMA_qdmaConfigArgs();`.

```
#define OUTPUT_START 0x80000000
...
#define MEMPORT_ADDR  ((volatile unsigned *)0x68000020)
#define MEMPORT_DATA  ((volatile unsigned *)0x68000030)
...

volatile Uint32 *imageptrout;
...

  imageptrout = (Uint32 *)OUTPUT_START;
  ...
  (*MEMPORT_ADDR) = ...;
  sec_out_len = ...;

  EDMA_qdmaConfigArgs(
    0x40380001,
   EDMA_SRC_OF(MEMPORT_DATA),
   EDMA_CNT_OF(sec_out_len),
   EDMA_DST_OF(imageptrout),
   EDMA_IDX_OF(0x00000004)
  );
  ...
```

První parametr je hodnota registru OPT, druhým parametrem je zdrojová adresa, třetí je délka přenášených dat, čtvrtý je cílová adresa a pátý parametr udává o kolik se mají adresy inkrementovat. V tomto případě je nastaven režim přenosu s fixní zdrojovou adresou, protože přistupujeme do paměťového portu, kde je adresa aktualizována automaticky. Cílová adresa je inkremetována, protože se zapisuje do již do konkrétní paměti - připojené SDRAM na modulu DX64 (viz Sekce 3). Přenos je inicializován s kódem rovným 8.

Registry paměťových portů jsou zpřístupněny přes ukazatele do paměťového prostoru rozhraní procesoru EMIFB, jak je vidět v předešlém příkladu zdrojového kódu.

Nastavení přerušení

Aby bylo možné efektivně spouštět a využívat EDMA přenosů dat je výhodné využít přerušení. Pro práci s přerušením lze využít také funkce knihovny CSL.

Následující příklad kódu ukazuje jak povolit jednotlivá přerušení, v tomto případě je to přerušení od EDMA řadiče a externího signálu EXT4. Pro přerušení EDMA řadičem je také nutné nastavit kód přenosu, který může přerušení vyvolat.

```
#define EDMA_COMPLETE_CODE 8
...

IRQ_resetAll();
IRQ_globalEnable();
IRQ_nmiEnable();
IRQ_enable(IRQ_EVT_EDMAINT);
EDMA_intEnable(EDMA_COMPLETE_CODE);
  IRQ_enable(IRQ_EVT_EXTINT4);
  ...
```

Pro obsluhu přerušení lze využít sadu funkcí, které jsou užity v následujícím příkladu obslužné rutiny pro přerušení od EDMA.

```
interrupt void IsrEdmaInt()
{
uint32_t gie = IRQ_globalDisable();
IRQ_clear(IRQ_EVT_EDMAINT);
EDMA_intClear(EDMA_COMPLETE_CODE);

  ...

IRQ_globalRestore(gie);
}
```

Aby se vykonala příslušná obslužná rutina, musí být v sekci `vectors` na dané adrese pro konkrétní přerušení uloženy instrukce pro skok na adresu obslužné rutiny.

6 Komunikace hostitelského systému s deskou Uni1P a moduly DX64

S deskou Uni1P a jejími moduly DX64 lze komunikovat pomocí implementovaných funkcí v systémovém FPGA. Veškerá komunikace s částmi desky Uni1P je realizována pomocí systémového FPGA, ve kterém je implementováno například "Host Port Interface" rozhraní a konfigurační rozhraní pro všechny FPGA na desce. Most mezi systémovým FPGA a hostitelským systémem tvoří PCI rozhraní implementované pomocí obvodu PLX 9054 podpořený ve vývojových nástrojích WinDriver.

6.1 Komunikace s modulem DX64 přes HPI

S moduly DX64 lze komunikovat přes rozhraní "Host Port Interface" (HPI) DSP, které je implementováno v systémovém FPGA. Mezi hostitelským systémem a adresním prostorem DSP na modulu DX64 lze komunikovat pomocí sady funkcí popsaných v jazyce C využívající volání funkcí z nástroje WinDriver. Funkce umožňují číst nebo zapisovat jednotlivá slova, bloky dat a nebo bloky dat s využitím DMA přenosů z celého adresního prostoru DSP. Pomocí HPI lze přistupovat nejen k vnitřní paměti DSP, ale i ke konfiguračním registrům jednotlivých periferií procesoru (EMIF) a do připojené paměti SDRAM.

Ke komunikaci mezi hostitelsým systémem a DSP na modulu DX64 je využita část vnitřní paměti DSP jako sdílená paměť pro předávání hodnot pro řízení programu.

Sdílená paměť je vyhrazena pomocí skriptu pro linkování programu pro DSP. Jednotlivé hodnoty parametrů, které se předávají, jsou předávány na adresy definované v hlavičkovém souboru. Hlavičkový soubor může být sdílen jak pro program v DSP, tak i pro program běžící na hostitelském systému, protože

funkce volané hostitelského systému využívají přes HPI tentýž adresní prostor. Jednotlivé funkce jsou popsány v kapitole D.

6.2 Přístup ke konfiguračnímu rozhraní obvodů FPGA

S využitím funkcí komunikujících s deskou Uni1P lze přistupovat ke konfiguračnímu rozhraní obvodů FPGA. Konfigurační rozhraní má dva registry, jeden řídící a jeden datový viz [4]. Datová propustnost konfiguračního rozharní je 25 MB/s.

Pro konfiguraci jednotlivých obvodů FPGA je třeba zapsat bitovou mapu, která určuje které rozhraní obvodů bude aktivní, do řídícího registru (pole DEVICE viz [4]). Na vstupu konfiguračního rozhraní je implementována vyrovnávací paměť (FIFO), jejíž využití lze přečíst z řídícího registru (pole COUNT viz [4]). Během nahrávání bitstreamů nesmí dojít k přetečení této paměti. Funkce pro zápis do adresového prostoru desky Uni1P jsou popsány v kapitole D.

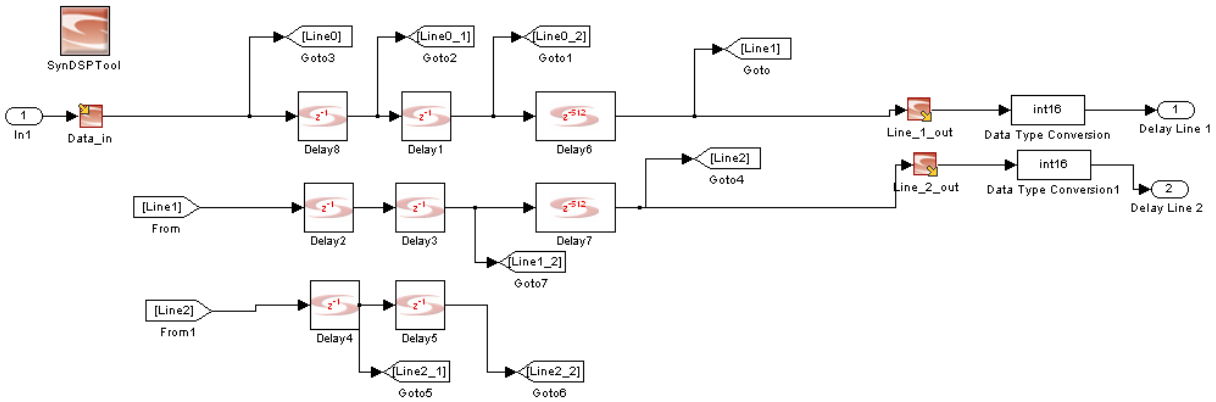
7 Implementace hranového detektoru

S využitím desky Uni1P a moduly DX64 byl implementován obrazový filtr s podporou částečné dynamické rekonfigurace. Jako obrazový filtr byl implementován Sobel užívaný k hranové detekci.

Filtr je implementován v obvodu FPGA na modulu DX64 a obrazová data jsou předávána pomocí připojeného DSP.

7.1 Návrh hranového detektoru pomocí nástroje Synplify DSP

Nástroj Synplify DSP je nadstavbou pro nástroj Simulik firmy MathWroks. Obrazový filtr je popsán schématickým propojením jednotlivých výpočetních bloků (viz obrázek 4). Výsledné propojení lze exportovat do jazyka pro popis hardware jako například jazyk VHDL. Během exportu do VHDL je provedena optimalizace vzhledem k cílové platformě (FPGA).



Obrázek 4: Příklad zapojení části hranového detektoru v Synplify DSP

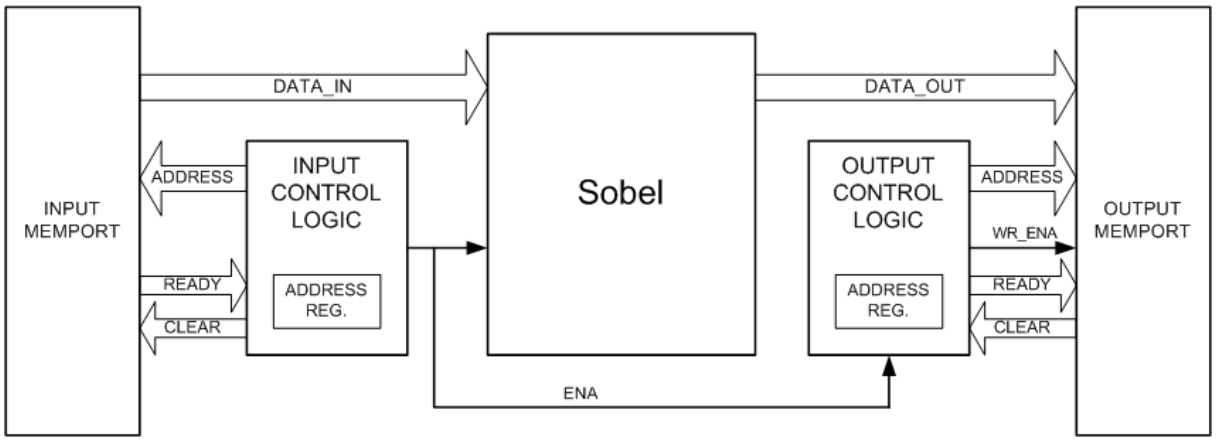
Model hranového detektoru je uložen na přiloženém médiu v adresáři `Sobel/`. Po otevření modelu jsou načteny parametry přiloženého obrazu `image.jpg`. Parametry obrazu jsou použity při návrhu a vygenerování hranového detektoru. Pro vygenerování hranového detektoru je třeba nejdříve spustit simulaci a poté detektor syntetizovat pomocí ”SynDSP Tool”. Po dokončení syntézy jsou vygenerovány VHDL soubory s implementací hranového detektoru, test a testovací data.

7.2 Propojení hranového detektoru a paměťových portů

Výsledkem exportu hranového detektoru je zdrojový kód komponenty detektoru ve VHDL, který bude instanciován v celkovém návrhu aplikace v FPGA. V tomto případě je třeba detektor připojit k paměťovým portům, pomocí nichž budou předávána obrazová data.

Jeden vstupní port bude využit pro vstupní data a druhý pro výstupní data. K hranovému detektoru je třeba připojit paměťová rozhraní portů a řídit jejich adresování. Adresování pamětí je určeno na základě signálů určující obsazenost sektorů paměťových portů a vlastní hodnotě čítače adres v rámci jednoho vstupního a jednoho výstupního sektoru. Každý sektor je po zpravování příslušným způsobem označen v registru MEMPORT_READY (registr MEMPORT_OVF není v této aplikaci využit), a tak může být informováno DSP o průběhu zpracování dat.

Na obrázku 5 je znázorněna implementace propojení filtru s paměťovými porty. Řídící logika na vstupu adresuje paměť vstupního portu od prvního (od nejnižší adresy) sektoru s platnými daty. Platnost dat v jednotlivých sektorech je určena pomocí signálu READY, který je odvozen od příslušných hodnot v registru MEMPORT_READY. Po zpracování jednoho vstupního sektoru je sektor označen jako prázdný pomocí signálu CLEAR a je adresován další sektor. Pokud žádný sektor není platný je detektor pozastaven pomocí signálu ENA. Obdobným způsobem je obsloužen výstup detektoru, který zapisuje na poslední volný sektor výstupního portu. Detektor má jistou latenci než vybaví data. Latence je kompenzována zpožděním signálu ENA (zpožděný signál WR_ENA), který povoluje zápis do paměti výstupního portu a její adresaci. Během zápisu jsou postupně označovány sektory s platnými daty. Řídící logika si udržuje informaci o všech sektorech, do kterých bylo zapsáno a předává tyto hodnoty do registru MEMPORT_READY pomocí signálu READY. Pokud dojde k zápisu do registru MEMPORT_READY ze strany DSP jsou tyto hodnoty smazány pomocí signálu CLEAR. Smazání informace o platnosti sektoru je provedeno zápisem logické 1 na příslušné pozici.



Obrázek 5: Znázornění propojení paměťových portů a hranového detektoru

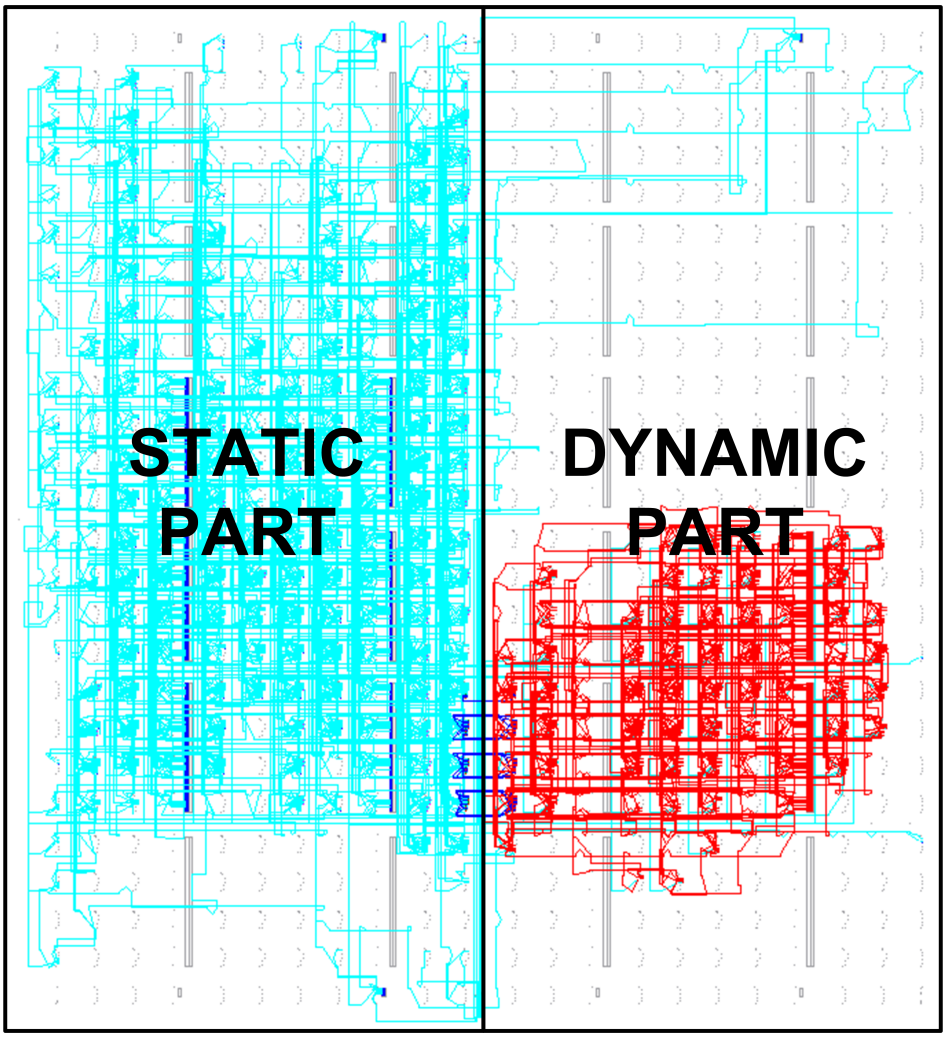
7.3 Využití částečné dynamické rekonfigurace

Pro efektivnější využití obvodů relativně malého FPGA na modulu DX64 se nabízí možnost použít částečnou dynamickou rekonfiguraci a měnit obrazové filtry za běhu aplikace. Část implementace obsahující paměťové porty a rozhraní EMIFB může být použita beze změny (statická část) a měněna může být pouze část s implementovaným obrazovým filtrem (dynamická část). Na obrázku 6 je pro ilustraci znázorněn grafický pohled nástroje ”FPGA editor” na implementaci filtru pro hranovou detekci.

Propojení mezi statickou a dynamickou částí tvoří datové rozhraní filtru, které se skládá z 8-mi bitového datového vstupu, 16-ti bitového datového výstupu, signálu určující platnost dat a resetovacího signálu.

Pokud je signál určující platnost dat aktivní, tak je každý hodinový takt načtena hodnota jednoho obrazového bodu. Platnost výstupních dat je dána (konstantní) latencí filtru.

Vstupní a výstupní data jsou předávána pomocí adresovací logiky ve statické části z/do adresového prostoru paměťových portů.



Obrázek 6: Grafický pohled nástroje ”FPGA editor” na implementaci filtru

7.4 Program pro DSP

Program DSP obsluhuje předávání dat ze sdílené paměti mezi DSP a hostitelským systémem. Komunikace mezi DSP a hostitelským systémem je řízena pomocí hodnot sdílených proměnných v paměti. Předávány jsou hodnoty adres, velikostí a platnosti vstupních a výstupních dat.

Na obrázku 7 je znázorněn vývojový graf hlavního programu a obslužných rutin přerušení. Hlavní program čeká na vstupní data a postupně je zpracovává v závislosti na počtu volných vstupních sektorů. Přenos dat do vstupních sektorů je inicializován pomocí řadiče EDMA. Po dokončení přenosu dat jsou označeny všechny zapsané sektory jako platné. Konec přenosu je indikován přerušením od EDMA řadiče. Obslužná rutina přerušení zkontroluje zda šlo o přenos vstupních a nebo výstupních dat. Pokud byla přenášena vstupní data, přestane hlavní program čekat na dokončení tohoto přenosu a pokračuje. Pokud šlo o přenos výstupních dat, jsou přenesené sektory uvolněny. Vstupní sektory jsou zpracovány pomocí hranového detektoru a výstupní data jsou postupně zapisována do výstupních sektorů, které jsou

označovány jako platné. DSP procesor je pomocí externího přerušení EXT4 informován o platných datech ve výstupních sektorech a je provedena obslužná rutina. Obslužná rutina pro signál EXT4 určí počet přenášených sektorů a jejich adresu a inicializuje EDMA přenos. Po dokončení přenosu je generováno přerušení EDMA řadiče a postupuje se tak, jak bylo popsáno výše.

Pokud byla zpracována všechna data, pak je informován nadřazený systém o dokončení zpracování jednoho obrazového snímku.

7.5 Program na straně PC

Program na straně osobního počítače (hostitelského systému) je psán v jazyce C. Pro komunikaci s deskou Uni1P využívá knihoven WinDriver, ve kterých je podpořena komunikace s obvodem PLX9054 tvořící rozhraní PCI rozhraní desky.

Program nejdříve inicializuje spojení s deskou Uni1P. Pro využití modulu DX64 je třeba nahrát program pro DSP daného modulu. Program je uložen v binárním souboru, viz příloha B.4. Před jeho zavedením musí být nakonfigurovány obvody FPGA na desce Uni1p a na daném modulu DX64, viz příloha B. Než je program zaveden, je DSP držen v resetovacím stavu, po jeho zavedení přes HPI rozhraní je z tohoto stavu uvolněn.

Obraz z kamery je snímán pomocí knihovny OpenCV. Je převeden na stupně šedi na velikost 1024 x 768 obrazových bodů, protože výchozí implementace hranového detektoru je navržena pro obraz s 1024 obrazovými sloupci. Obrazová data jsou nahrána do sdílené paměti DSP na daném modulu DX64 a data jsou zpracována viz sekce 7.2 a 7.4. Na obrázku 8 je znázorněn vývojový diagram programu běžícího na straně osobního počítače.

Každý obrazový bod vstupních dat je reprezentován jedním bajtem a pro výstupní data dvěmi bajty. Výstupní data musí být na straně PC normalizována a poté jsou použita k zobrazení obrazu. Hranový detektor lze za běhu programu rekonfigurovat. Další implementace hranového detektoru jsou implantovány pro obraz s 512 a 752 obrazovými sloupci. Pokud dojde k rekonfiguraci je pro hranovou detekci využit pouze výřez z obrazu. Pozici výřezu lze řídit pomocí klávesnice viz tabulka 3.

Klávesa	Význam klávesy
A	Posun okna výřezu doleva o 10 obrazových bodů.
S	Posun okna výřezu dolu o 10 obrazových bodů.
D	Posun okna výřezu doprava o 10 obrazových bodů.
W	Posun okna výřezu nahoru o 10 obrazových bodů.
+	Zvýšení okna výřezu o 10 obrazových bodů.
-	Snížení okna výřezu o 10 obrazových bodů.
0..9	Volba bitstreamu pro rekonfiguraci.
ESC	Ukončení aplikace.

Tabulka 3: Ovládání aplikace pomocí klávesnice.

Program je spouštěn pomocí dávky, která inicializuje systémové FPGA na desce Uni1P a FPGA na modulech DX64 (viz B.2) a spustí program na straně PC. Program pro PC komunikuje s modulem M2 (třetí pozice na desce Uni1P) a spouští se následovně:

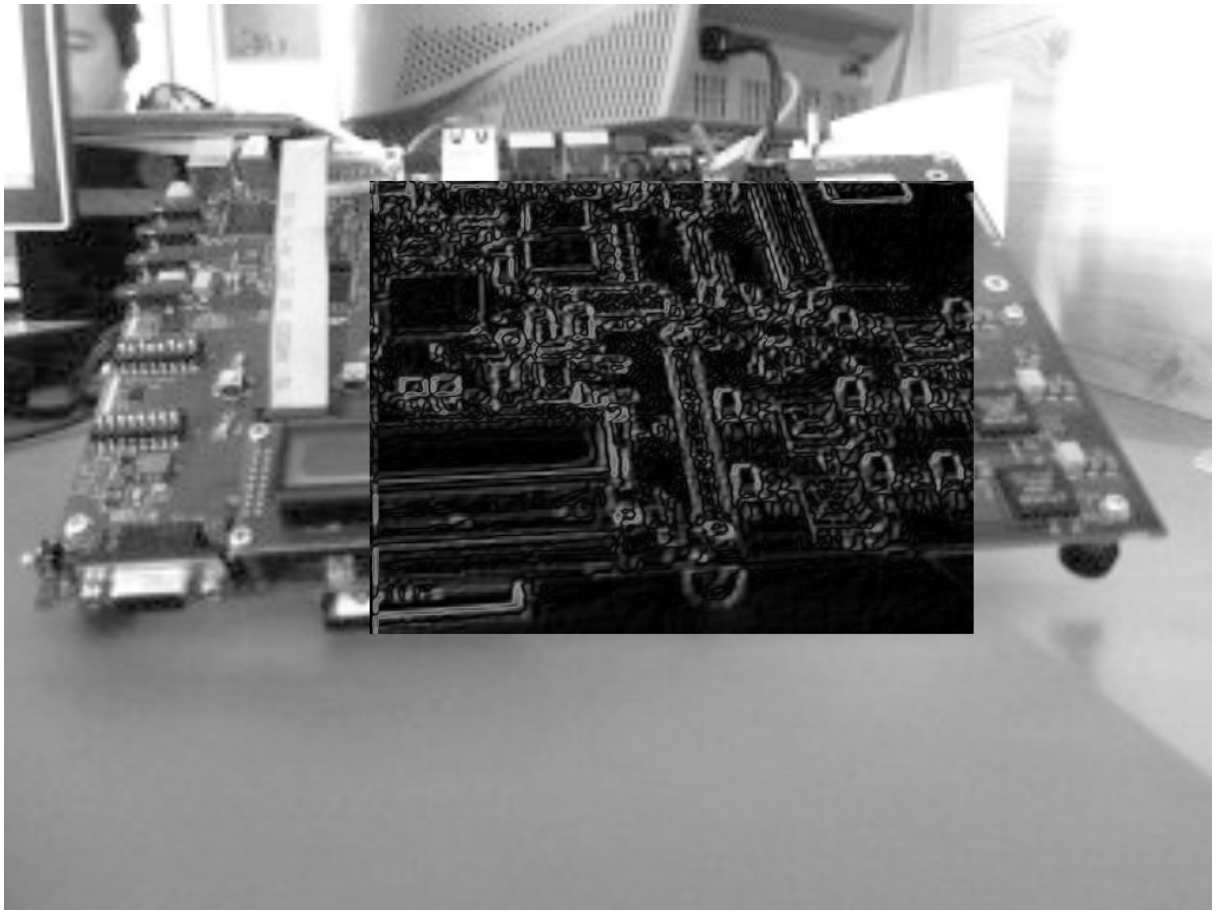
```
sobel.exe <dsp_binary_file> [[<bitstream_0> <width_0>] ... [<bitstream_9> <width_9>]]
```

Parametr `<dsp_binary_file>` je cesta k binárnímu souboru obsahující program pro DSP na modulu DX64. Následujícími parametry může být seznam (částečných) bitstreamů filtrů a k nim příslušných šířek v obrazových bodech, pro jaké byly jednotlivé filtry implementovány. Maximální počet parametrů pro jednotlivé bitstreamy je omezen na deset. Seznam bitstreamů může obsahovat celkové nebo částečné bitstreamy. Částečné bitstreamy lze použít za předpokladu, že byl předem nahrán celkový bitstream (například v rámci inicializace desky).

7.6 Výsledky

V tomto dokumentu byla popsána architektura desky Uni1P a modulů DX64 a návrhové postupy při vytváření aplikací na nich běžící. Jako příklad aplikace byla zvolena implementace filtru pro hranovou detekci obrazu. Příklad obrazového výstupu hranové detekce z webové kamery je na obrázku 9. Snímková frekvence je závislá na velikosti obrazu nebo jeho výřezu, který je filtrován. Pro obraz o velikosti 1024 x 768 obrazových bodů je snímková frekvence 1,6 snímku za sekundu. Nízká propustnost komunikace s modulem DX64 a PC způsobuje podstatné zpoždění při zpracovávání jednotlivých snímků. Propustnost zápisu do paměťového prostoru modulu DX64 je 2,5 MB/s a propustnost čtení je 0,8 MB/s.

Modul DX64 čte a zpracovává vstupní obrazová data rychlostí 50 MB/s a výsledky ukládá rychlostí 100 MB/s, protože každému vstupnímu bytu (obrazovému pixelu) odpovídá jedno výstupní 16-ti bitové slovo.



Obrázek 9: Příklad výřezu obrazu s hranovou detekcí.

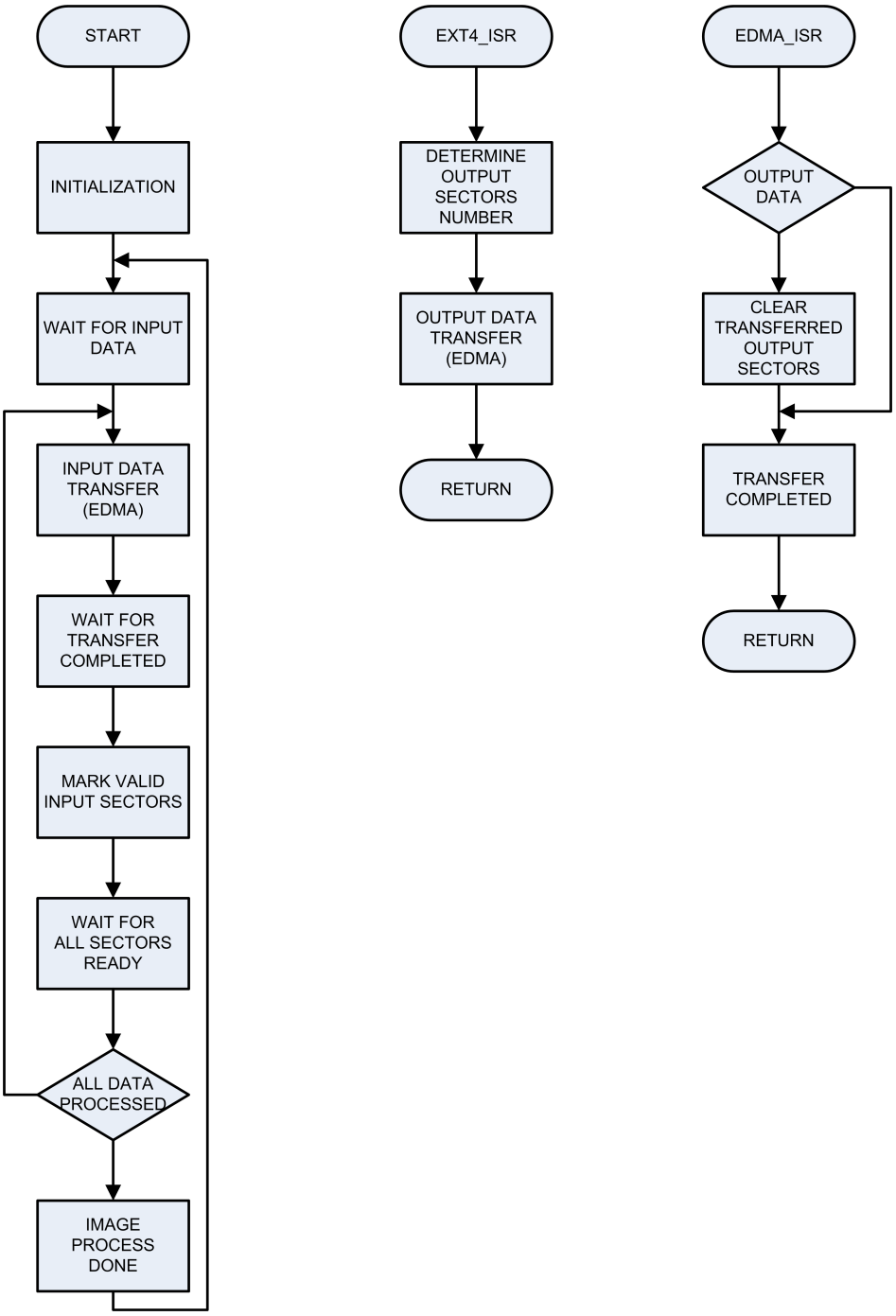
V aplikaci je využita částečná dynamická rekonfigurace, která je použita pro změnu funkce hranového detektoru. Struktura implementace hranového detektoru je závislá na šířce zpracovávaného obrazu. Vzhledem k efektivnímu využití zdrojů obvodu FPGA je hranový detektor implementován v několika modifikacích pro dané rozměry obrazu a za běhu aplikace je propojení obvodu FPGA měněno. S využitím částečné dynamické rekonfigurace je dosaženo kratší doby potřebné pro změnu funkce obvodu, velikost paměti potřebné pro uložení bitstreamu a navíc může být část funkce obvodu zachována ve statické části. Velikost bitstreamů pro jednotlivé implementace hranového detektoru a potřebná doba pro rekonfiguraci je uvedena v tabulce 4. Úspora paměti pro uložení bitstreamu i doby rekonfigurace se v našem případě pohybuje okolo 68 %.

Bitstream	Velikost bitstreamu	Doba rekonfigurace
Plný bitstream	199 204 B	7,97 ms
Částečný pro 1024 px	62 672 B	2,51 ms
Částečný pro 752 px	62 600 B	2,49 ms
Částečný pro 512 px	62 252 B	2,50 ms

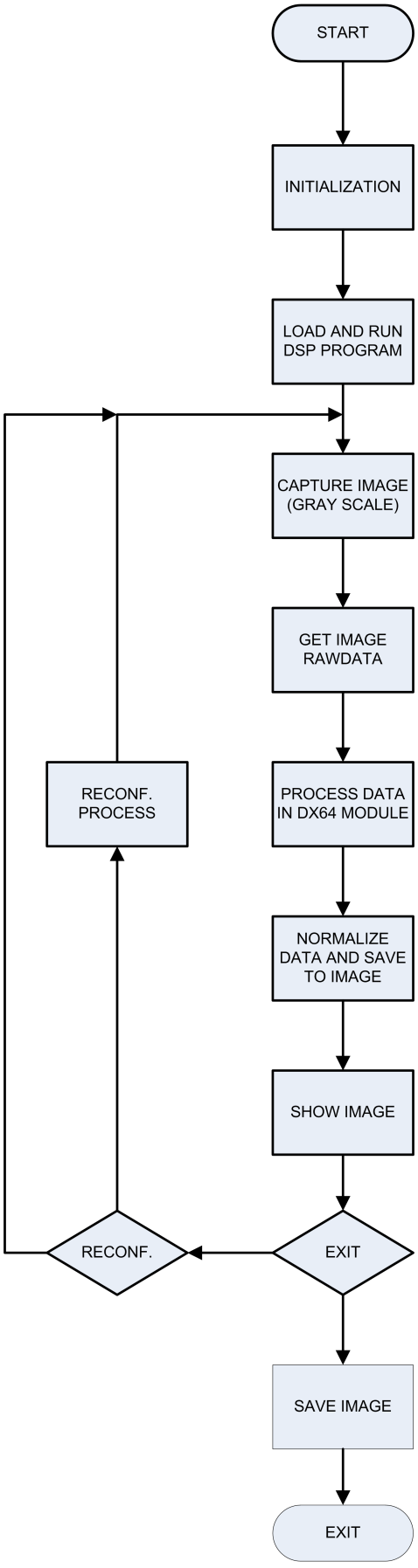
Tabulka 4: Výsledky úspor při využití částečné dynamické rekonfigurace

8 Poděkování

Tato zpráva vznikla za podpory grantu AVČR č. 1ET400750408 – Rapid prototyping tools for development of HW-accelerated embedded image- and video-processing applications.



Obrázek 7: Vývojový diagram programu pro DSP.



Obrázek 8: Vývojový diagram programu pro osobního počítače.

9 Výpis CDROM

Na CD se nachází text dokumentu, zdrojové kódy pro programy DSP a PC, model hranového detektoru v Synplify DSP a VHDL kódy pro obvody FPGA.

Přiložené CD má následující adresářovou strukturu:

```
.
|-- boot_sobel/                  Alikace pro hranovou detekci
|   |-- bin/                     Bitstreamy pro obvody FPGA
|   |-- common/                  Společné soubory pro program DSP a PC
|   |-- dsp/                     Zdrojové kódy programu pro DSP
|   |-- fw_pr/                   Nástroje a zdrojové kódy pro PR
|   |   `-- PR_Flow/
|   |       `-- synth/           VHDL zdrojové kódy
|   |           |-- common/      Zdrojové kódy knihoven
|   |           |-- rmodule_0/   Zdrojové kódy pro Sobel (1024 px)
|   |           |-- rmodule_1/   Zdrojové kódy pro Sobel (512 px)
|   |           |-- rmodule_2/   Zdrojové kódy pro Sobel (752 px)
|   |           |-- static/      Zdrojové kódy statické části
|   |           `-- top/         Zdrojové kódy pro nejvyšší úovně návthu
|   |-- HPIheader/               Definice paměťového prostoru HPI
|   |-- PC_part/                 Program pro PC
|   |   |-- lib                  Potřebné knihovny
|   |   `-- src                  Zdrojové kódy programu
|   |-- utils/                   Nástroje pro komunikaci s deskou Uni1P
|   `-- sobel.bat                Dávka pro spuštění aplikace
|-- doc/                         text dokumentu ve formátu PDF
|-- Sobel/                       Model hranového detektoru - Synplify DSP
`-- readme.txt
```

Reference

[1] Jungo Ltd. *WinDriver PCI for Windows Development Tool* [online]. Dostupné na WWW: `http://www.jungo.com/`

[2] Xilinx. *Virtex-II Platform FPGA User Guide* [online]. Dostupné na WWW: `http://www.xilinx.com/support/documentation/user_guides/ug002.pdf`

[3] Xilinx. *Virtex FPGA Series Configuration and Readback* [online]. Dostupné na WWW: `http://www.xilinx.com/support/documentation/application_notes/xapp138.pdf`

[4] Antonín Hegar. *Uni1P - Registry*.

[5] Texas Instruments. *TMS320C6414, TMS320C6415, TMS320C6416 FIXED-POINT DIGITAL SIGNAL PROCESSORS* [online]. Dostupné na WWW: `http://focus.ti.com/lit/ds/sprs146n/sprs146n.pdf`

[6] Texas Instruments. *TMS320C6000 Peripherals Reference Guide* [online]. Dostupné na WWW: `http://archer.ee.nctu.edu.tw/class/dsplab/07f/..%5C07s%5Creference%5Cspru190d.pdf`

A Generování bitstreamu

Všechny obvody FPGA osazené na Uni1P a modulech DX64 sdílejí konfigurační rozhraní - tzv. ”Slave SelectMAP” (viz 2). Konfigurační rozhraní je řízeno systémovým FPGA.

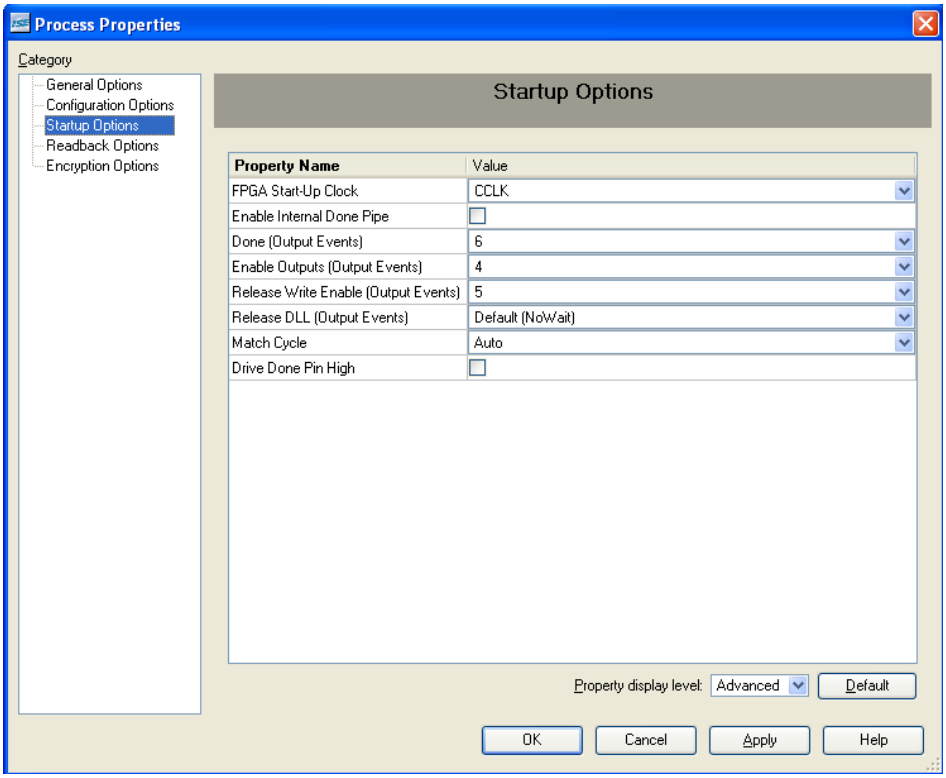
Pokud chceme nakonfigurovat nějaký obvod FPGA musíme aktivovat PROG_B. Signál PROG_B je sdílený pro všechny obvody, proto přejdou všechny obvody FPGA do stavu čekající na nahrání konfiguračního bitstreamu. Příslušným signálem CS_B zvolíme cílový obvod a nahrajeme bitstream viz [2]. Po nakonfigurování je obvodem FPGA standardně aktivován signál DONE. V našem případě nechceme aby byl signál DONE aktivován, protože by sdílený aktivní signál DONE ukončil konfigurační režim i ostatním obvodům - signál DONE je implementován technologii ”open collector”.

Každý obvod FPGA obsahuje automat, který řídí jeho konfiguraci a generuje signály pro jeho řízení - tedy i signál DONE [3]. Pomocí parametrů nástroje `bitgen` lze chování výstupů konfiguračního automatu ovlivnit.

Pro správné vygenerování bitstreamu pro modul DX64 je zapotřebí nastavit v projektu ISE několik parametrů pro generování konfiguračních souborů.

Ve vlastnostech ”Generate Programming File” je zapotřebí v záložce ”Startup Options” nastavit tyto parametry:

- Done (Output Events) na hodnotu 6
- Enable Outputs (Output Events) na hodnotu 4
- Release Write Enable (Output Events) na hodnotu 5



Obrázek 10: Obr

Pokud spouštíme `bitgen` z příkazové řádky, můžeme parametry předat následovně:

```
bitgen -w -g DONE_cycle:6 -g GWE_cycle:5 -g GTS_cycle:4 <NCD File>
```

Výsledný binární soubor pro aplikaci konfigurující obvod FPGA je převeden pomocí příkazu `promgen`.

```
promgen -w -p bin -c FF -o <output file name> -u 0 <input file name>.bit -x xc1702l
```

B Nahrání bitstremů do jednotlivých FPGA a programu pro DSP

B.1 Konfigurace Systémového FPGA

Bitstream pro systémové FPGA lze nahrát pomocí aplikace `uni1p.exe` následující dávkou:

```
uni1p.exe -byte -addr 0xe0000000 0xe
uni1p.exe -byte -addr 0xe0000000 0xf
uni1p.exe -byte -addr 0xe0000000 0x1
uni1p.exe -byte -addr 0xe0000008 -fix <binary file>
uni1p.exe -byte -addr 0xe0000000 0xf
```

Aplikace nejdříve zapíše do konfiguračního registru (adresa 0xe000000) rozhraní SelectMAP, které je implementováno v obvodu CPLD desky Uni1P. Datový registr rozhraní SelectMAP je namapováno na adresu 0xe0000008.

Bližší specifikace implementace rozhraní SelectMAP je uvedena v [4].

Použití apliace `uni1p.exe` je následující:

```
uni1p.exe [-reset] [-addr <ADDR>] [-dma | -reg] [-dword | -byte] [-fix]
          [-length <LEN>] [file | data ]
```

Význam přepínačů je následující:

Přepínač	Význam přepínače
reset	Reset obvodu PLX
addr	Počáteční adresa pro datový přístup.
dma	Využití DMA při datových přenosech.
reg	Přístup do registrů obvodu PLX.
fix	Fixní mód adresace.
length	Délka (LEN) čtených dat.

Tabulka 5: Význam přepínačů aplikace uni1p_fpga.exe

B.2 Konfigurace obvodů FPGA

Všechny obvody FPGA osazené na Uni1P nebo na modulech DX64 lze konfigurovat pomocí aplikace `uni1p_fpga.exe`. Programování obvodů FPGA je aktivováno signálem PROG_B - to odpovídá volání aplikace s přepínačem `clear`. Výběr obvodu, který bude konfigurován, je určen pomocí příslušných přepínačů - viz tabulka 6. Obvody lze konfigurovat současně, protože jednotlivé přepínače aktivují odpovídající signály CS obvodů FPGA - nabízí se tak například konfigurace všech FPGA na modulech DX64 současně.

Následující dávka ukazuje příklad, jak nahrát bitstream do Master FPGA a současně do modulů 2 a 3:

```
uni1p_fpga.exe -clear
uni1p_fpga.exe -master <binary file>
uni1p_fpga.exe -m2 -m3 <binary file>
```

Přepínač	Význam přepínače
-clear	Přepínač aktivuje signál PROG_B.
-reset	Přepínač aktivuje resetovací signál pro vybrané obvody FPGA.
-master	Výběr obvodu Master FPGA.
-m0	Výběr obvodu na modulu 0 - DX64
-m1	Výběr obvodu na modulu 1 - DX64
-m2	Výběr obvodu na modulu 2 - DX64
-m3	Výběr obvodu na modulu 3 - DX64
-all	Současný výběr obvodů na modulech 0-3 - DX64

Tabulka 6: Význam přepínačů aplikace uni1p_fpga.exe

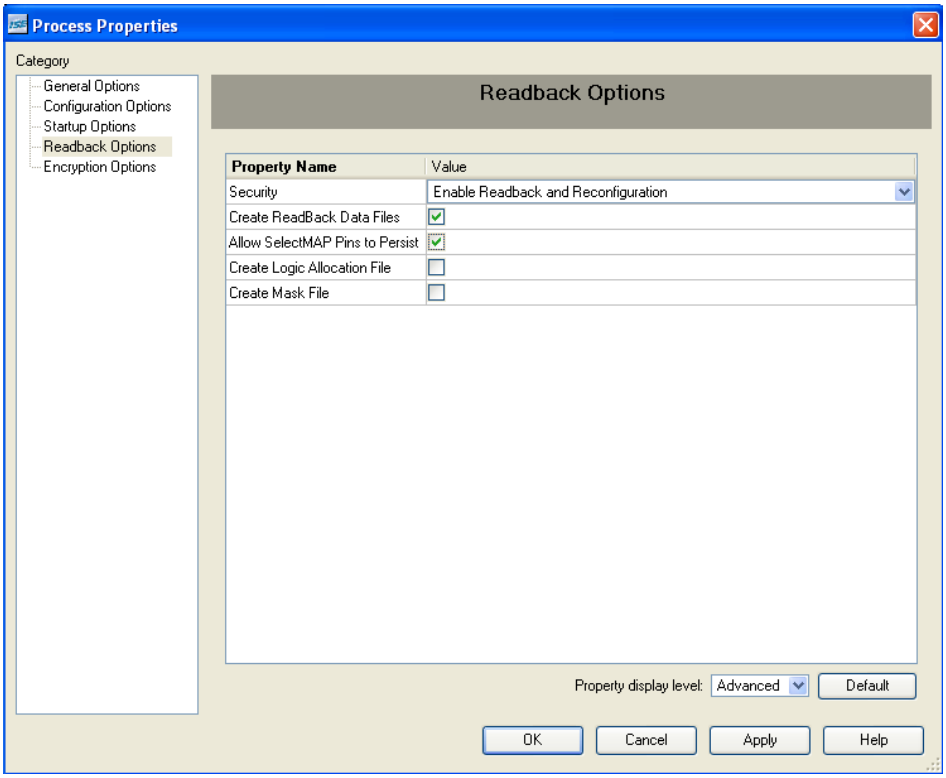
Použití apliace `uni1p_fpga.exe` je následující:

```
uni1p_fpga.exe  [-clear] [-master | -comm | -m0 | -m1 | -m2 | -m3 | -all]
                [-reset | <file.bin>]
```

B.3 Dynamická rekonfigurace na Uni1P

Při generování jednotlivých bitstreamů pro částečnou dynamickou rekonfiguraci je třeba předat nástrojům příslušný parametr, specifikovaný v příloze A a parametr, který zajistí aby konfigurační rozhraní obvodu FPGA bylo stále aktivní (parametr `Persist`). V ISE lze parametr nastavit v kontextovém menu pro generování bitstreamu, jak je vidět na obrázku 11 nebo pomocí příkazové řádky následovně:

```
bitgen -w -g DONE_cycle:6 -g GWE_cycle:5 -g GTS_cycle:4 -g Persist:Yes <NCD File>
```



Obrázek 11: Menu parametrů pro generování bitstreamu

Pokud jsou pro generování částečných bitstremů použity skripty `PR_verifydesigan.pl` a `PR_assemble.pl` popsané v jazyce Perl, je třeba zkontrolovat a případně doplnit parametry viz A do seznamu podporovaných přepínačů. Příslušná část skriptu by měla vypadat například takto:

```
our %_validFlags = ("-g compress"   => "-g Compress",
                    "-g encrypt"    => "-g Encrypt",
                    "-g persist"    => "-g Persist",
                    "-g keyfile"    => "-g KeyFile",
                    "-g binary"     => "-g Binary",
                    "-g crc"        => "-g CRC",
                    "-g done_cycle" => "-g DONE_cycle",
                    "-g gts_cycle"  => "-g GTS_cycle",
                    "-g gwe_cycle"  => "-g GWE_cycle",
                    "-d"            => "-d",
                    "-w"            => "-w",
                  );
```

Plný bitstream lze nahrát způsobem, popsaným v příloze B.2, poté je možné nahrávat částečné nebo i plné bitstreamy dynamicky, protože rozhraní SelectMAP je stále aktivní. Postup je tedy stejný jako v příloze B.2, ale nesmí být aktivován signál PROG_B, tedy nepoužijeme přepínač `clear`. Vývody konfiguračního rozhraní obvodu FPGA nemohou být použity pro jiný účel, pokud je konfigurační rozhraní příslušným parametrem nastaveno jako aktivní.

B.4 Generování binárního souboru programu pro DSP a jeho zavedení

Po přeložení programu pro DSP získáme OUT soubor, který převedeme pomocí aplikace `hex6x.exe` na HEX soubor. Aplikaci `hex6x.exe` předám soubor s definicí vstupního souboru, výstupního souboru a formátu paměti. Příklad souboru s příkazy pro `hex6x.exe` je následující:

```
<infile.out>
-i
-memwidth 8
ROMS
{
  FLASH: org = 0x00000000, len = 0x8000, romwidth = 8, files = {<outfile.hex>}
}
```

HEX soubor převedeme na binární soubor pomocí aplikace `hexbin2.exe` s následujícími parametry:

```
hexbin2.exe <infile.hex> <outfile.bin> I 0000 FFFF I
```

Binární soubor obsahující program pro DSP lze nahrát pomocí aplikace `uni1p_dsp.exe`.
Použití apliace `uni1p_dsp.exe` je následující:

```
uni1p_dsp.exe [-m0 | -m1 | -m2 | -m3] [-addr <addr>] <file.bin>
```

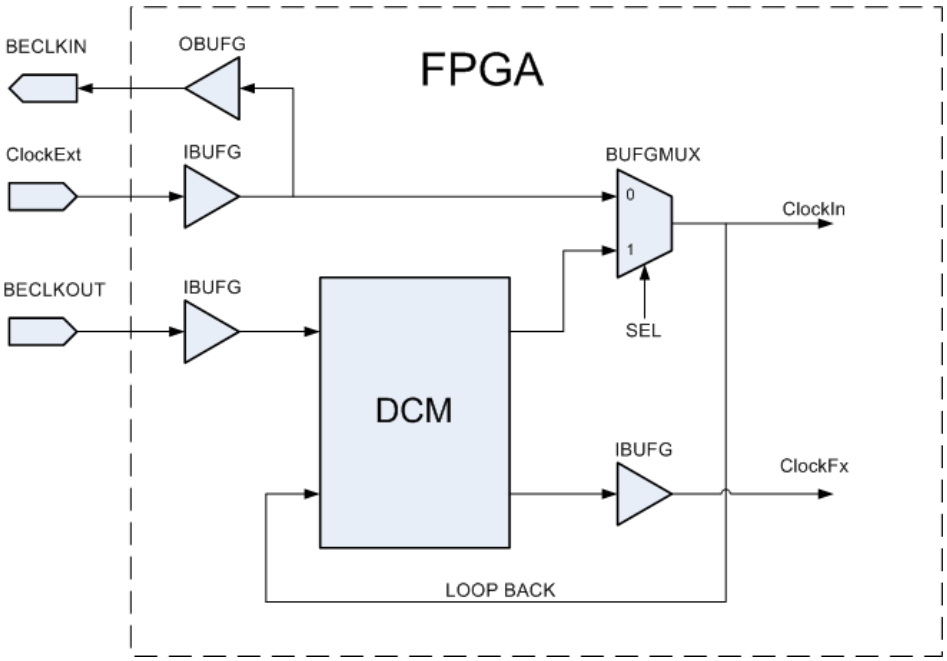
C Výběr zdroje hodinového signálu pro FPGA na DX64

Hodinový signál pro obvod FPGA je generován na výstupu BECLKOUT obvodu DSP. Frekvence hodinového signálu na výstupu BECLKOUT je dána násobičem základní frekvence (50 MHz). Hodnota násobiče je přímo určena logickými hodnotami na vstupech DSP (CLKMODE[1:0] viz [5]), které jsou připojeny k FPGA, a tak lze frekvenci určit pomocí propojení v FPGA.

Během resetu DSP není na výstupu BECLKOUT generován hodinový signál a proto kombinační logika přepne zdroj hodinového signálu na oscilátor na modulu DX64 (50 MHz). Způsob přepínání hodinového signálu v FPGA je znázorněn na obrázku 12.

V návrhu jsou pomocí obvodu FPGA nastaveny signály CLKMODE[1:0] na hodnotu 01b, což odpovídá vynásobení hodinového vstupu hodnotou šest. Výsledná pracovní frekvence procesoru je tedy 300 MHz. Vyšší násobič nelze použít, protože obvod osazený na modulu DX64 je navrhnut pro pracovní frekvenci do 500 MHz.

Hodinovým signálem pro FPGA je hodinový výstup rozhraní EMIFB. Frekvence hodinového signálu EMIFB je odvozena od pracovní frekvence procesoru podle konkrétního zapojení obvodu (viz [5]). Pro obvod zapojeném na modulu DX64 je frekvence rovna jedné šestině pracovní frekvence procesoru, tedy 50 MHz.



Obrázek 12: Způsob implementace hodinového rozvodu pro FPGA osazeném na DX64

D Funkce knihovny libUni1P

V této kapitole sou popsány jednotlivé funkce knihovny libUni1P, které jsou využívány pro komunikaci s deskou Uni1P a jejími moduly DX64.

```
int Uni1pInit(void);
```

Inicializace knihovny pro podporu komunikace s deskou Uni1P a moduly DX64.

```
UNI1P_HANDLE Uni1pOpenBoard(int slot);
```

Naváže komunikaci s kartou v daném slotu a vrací handler k desce Uni1P.

`slot` Číslo slotu, ve kterém deska Uni1P je.

```
int Uni1pHpiInit(UNI1P_HANDLE hPlx);
```

Inicializuje podporu pro komunikaci přes HPI.

`hPlx` Handler pro komunikaci s deskou Uni1P.

```
int Uni1pHpiDspReset(UNI1P_HANDLE hPlx,
                     tUni1pHpiModule dsp);
```

Vyvolá reset signálového procesoru na daném modulu DX64.

`hPlx` Handler pro komunikaci s deskou Uni1P.
`dsp` Určení modulu, na kterém se má DSP resetovat.

```
int Uni1pHpiDspRun(UNI1P_HANDLE hPlx,
                   tUni1pHpiModule dsp);
```

Uvolnění DSP z resetu a spuštění běhu programu.

`hPlx` Handler pro komunikaci s deskou Uni1P.
`dsp` Určení modulu, na kterém se má DSP resetovat.

```
int Uni1pHpiBlock(UNI1P_HANDLE hPlx,
                  BOOL read,
                  tUni1pHpiModule dsp,
                  DWORD addr,
                  int size,
                  DWORD *buff );
```

Funkce pro čtení/zápis dat z/do paměti do/z adresního prostoru DSP.

`hPlx` Handler pro komunikaci s deskou Uni1P.
`read` Parametr určuje zda se bude provádět čtení nebo zápis dat přes HPI.
`dsp` Určení modulu na desce Uni1P.
`addr` Adresa v rámci paměťového prostoru DSP.
`size` Velikost dat v DWORD
`buff` Ukazatel do paměti hostitelského systému.

```
int Uni1pHpiDma(UNI1P_HANDLE hPlx,
                BOOL read,
                tUni1pHpiModule dsp,
                DWORD addr,
                int size,
                DWORD *buff );
```

Funkce pro čtení/zápis dat z/do paměti do/z adresního prostoru DSP používající DMA přenosů.

`hPlx` Handler pro komunikaci s deskou Uni1P.
`read` Parametr určuje zda se bude provádět čtení nebo zápis dat přes HPI.
`dsp` Určení modulu na desce Uni1P.
`addr` Adresa v rámci paměťového prostoru DSP.
`size` Velikost dat v DWORD
`buff` Ukazatel do paměti hostitelského systému.

```
void Uni1pWriteDWord(UNI1P_HANDLE hPlx,
                     DWORD addr,
                     DWORD data);
```

Funkce zapisuje na danou adresu paměťového prostoru desky Uni1P dvojslovo.

`hPlx` Handler pro komunikaci s deskou Uni1P.
`addr` Adresa v rámci paměťového prostoru desky Uni1P.
`data` Hodnota dat pro zápis.

```
DWORD Uni1pReadDWord(UNI1P_HANDLE hPlx,
                     DWORD addr;
```

Funkce vrací hodnotu dvojslova přečteného z dané adresy paměťového prostoru desky Uni1P.

`hPlx` Handler pro komunikaci s deskou Uni1P.
`addr` Adresa v rámci paměťového prostoru desky Uni1P.

E Aplikační poznámky

Vlastnost konfiguračního rozhraní obvodu Virtex-II

Během implementace aplikací s využitím částečné dynamické rekonfigurace na obvodu Virtex-II docházelo náhodně k nesprávného nahrání bitstreamu a nedošlo k rekonfiguraci. Pro spolehlivé nahrání bitstreamu je nutné přidat na konec bitstreamu navíc jedno slovo pro konfigurační rozhraní. Jako dodatečné slovo je použito řídící slovo konfiguračního rozhraní NOP (No Operation), jehož hexadecimální hodnota je 0x20000000h.

Nastavení adresového registru paměťového portu

Pokud nastavujeme adresní registr paměťových portů před EDMA přenosem, může dojít k nespolehlivému přenosu dat. Experimentálně bylo ověřeno, že pokud po nastavení hodnoty adresního registru následuje jakékoliv čtení registrů paměťových portů a poté následuje EDMA přenos, data jsou přenesena korektně.

Změna komponenty EMIFB pro dynamickou rekonfiguraci

Během modulárního návrhu vhodného pro částečnou dynamickou rekonfiguraci je nutné znát dopředu umístění budičů hodinových rozvodů a musí bít instaciovány v nejvyšší úrovni návrhu (”top”). Umístění budičů hodinového rozvodu je určeno v souboru s definicemi rozmístění a časování návrhu v FPGA (UCF - ”User Constraint File”).

Jelikož komponenta pro rozhraní EMIFB obsahovala instanci budiče hodinového signálu, musela být přesunuta do vyšší úrovně v modulárním návrhu.