Moderní metody zpracování obrazu strukturou FPGA

Soběslav Valach

Ústav automatizace a měřicí techniky, FEKT VUT v Brně, Kolejní 2906/4, 612 00, Brno, valach@feec.vutbr.cz

Abstract:

Článek popisuje možnosti a metody návrhu systémů pro zpracování obrazových dat využívající struktury FPGA a zaměřuje se na rychlou a jednoduchou implementaci softwarového a firmwarového řešení doplněného o praktické výsledky.

1. Úvod

Moderní metody strojového vnímání mají specifické požadavky na získávání a zpracování dat. Současné metody jsou dobře propracované jak z hlediska snímací techniky (kamerové čipy, objektivy, přenos a ukládání dat), tak i vlastní algoritmizace. Obecnou nevýhodou je nutnost zpracovat velký objem dat a to co možná za nejkratší časový úsek.

Klasické přístupy využívají struktury, nebo systémy založené na kombinaci kamera a vyhodnocovací systém - obvykle v podobě platformy PC AT nebo signálových procesorů, kde se provádí algoritmus předepsaný programátorem. Výhodou takto koncipovaných řešení je bezesporu snadná dostupnost a velká míra flexibility řešení. Na druhé straně je nutné zmínit některé nevýhody výše uvedené koncepce, které znemožňují praktické nasazení v některých aplikacích, kde je třeba zpracovávat velké množství obrazových dat v reálném čase, při velmi nízké spotřebě energie a omezeném prostoru celého řešení.

2. Metody řešení

Zpracování výpočetně a datově náročných úloh se obvykle provádí na výkonných multi-procesorových počítačích, systémech řazených do clusterů, obvodech typu ASIC, DSP, systolických sítích nebo programovatelných strukturách typu FPGA a FPOA.

V článku se dále pouze zaměříme na systémy využívající programovatelné struktury typu FPGA. V první řadě si řekněme, co nám nabízí programovatelné struktury hradlových polí a jaké jejich vlastností můžeme použít v systémech vyhodnocujících obrazovou informaci.

Tabulka 1: Porovnání výhod a nevýhod řešení založených na FPGA

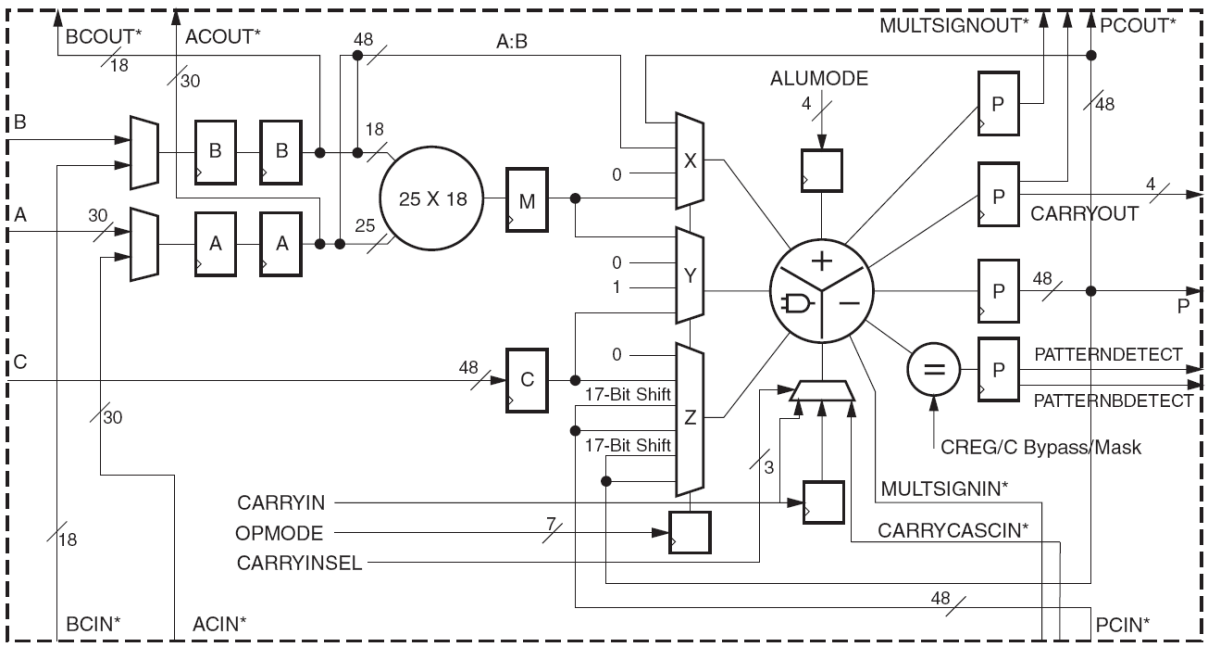
Výhody	Nevýhody
100% uživatelský návrh řešení	Znalost speciálních nástrojů
Vysoký výpočetní výkon	Časová náročnost návrhu řešení
Paralelní zpracování úlohy	Zdlouhavá verifikace, existují úlohy, které není možno paralelizovat
Obvykle nižší spotřeba energie	
Obvykle nižší cena řešení	Drahé vývojové nástroje
Kompaktní řešení	Vyšší cena vývoje
Množství vstupů a výstupů	
Rychlé interní pamětí	Omezená velikost interních paměťových bloků
Podpora DSP operací	Absence floatových jednotek
Clock management	
SoftCore procesory	Nižší výkon soft jader
Integrovaná hard jádra procesorů	Nižší výkon hard jader než u klasických řešení

Z tabulky 1. vyplývá, že návrh systému založený na strukturách FPGA poskytuje prakticky neomezené možnosti ale za cenu využití nástrojů a znalostí, které běžně

programátor ve své praxi nepoužívá. V případě, kdy nelze zvolený algoritmus paralelizovat je implementace architekturou FPGA nevhodná.

Ve své podstatě se jedná o návrh paralelní struktury, využívající jednotek integrovaných v FPGA rozšířených o další funkcionalitu pomocí propojovacích sítí, logických bloků a klopných obvodů.

Jednou z hlavních jednotek, používaných v oblastech zpracování signálu a tedy i zpracování obrazu jsou DSP bloky. Příklad DSP bloku integrovaného v hradlovém poli řady Virtex 5 je uveden na obrázku 1.



Obr. 1: DSP48E blok hradlového pole Virtex 5

DSP blok s označením DSP48E nabízí tyto základní funkce:

- násobičku 25 x 18 bitů
- MACC operace
- dynamický výběr operací v ALU jednotce (součet, rozdíl, logické operace)
- kaskádní řazení DSP bloků s podporou přenosu výsledků až 96bitů
- SIMD operace 4 x 12 bitů nebo 2 x 24 bitů v jednom DSP48E bloku
- 48-bitovou logickou jednotku
- barrel shifter
- rychlý 48-bitový čítač
- detekce vzorů
- zaokrouhlování výsledků, saturace, přetečení a podtečení
- podpora multiplexování dat v čase

V hradlovém poli Virtex 5 může DSP blok pracovat do frekvence 550MHz, přičemž počet jednotek v jednom pouzdru se pohybuje od desítek až po 1056 bloků.

Z výše uvedeného stručného popisu vyplývá, že FPGA je schopno řešit výpočetně náročné úlohy rychleji než klasické procesorové systémy. Na druhé straně je jasné, že výhody použití těchto struktur jsou omezeny vývojovými nástroji a schopnostmi návrháře cílové platformy.

Návrhy a vývoj komponent, které budou zpracovávat obrazová data musí zohlednit fakt, že hradlové pole diky masivnímu paralelismu, nebude mít možnost přistupovat do paměti v neomezeném rozsahu jako klasické procesory. Zpracování dat bude probíhat proudově v pipeline jednotkách, obvykle bez možnosti vrátit se zpět a vidět mezivýsledky nebo výsledky algoritmu.

Návrh vlastních komponent vychází z možností architektury FPGA. Jedná se především o využití vnitřních zdrojů (blokové pamětí, DSP bloky, logické bloky), stanovení vhodného časování a pochopitelně z omezení prostorových nároků uvnitř čipu na požadovanou funkci.

Pro tvorbu vlastního designu se obvykle používají programovací jazyky typu HDL (VHDL, Verilog). Jazyky HDL neposkytují návrháři dostatečnou formu abstrakce pro takto složité algoritmy, což značně komplikuje práci a snižuje flexibilitu řešení, kdy i jednoduché změny a jejich ověření zaberou nemálo času. Výstupem HDL jazyka je zápis v RTL formě, který je výborný pro verifikaci jednoduchých struktur typu čítač, multiplexer, stavový automat ale zcela nevhodný pro verifikace algoritmů zpracovávajících obrazovou informaci.

Oblast verifikace výsledků se řeší pomocí simulací na testovacích obrazcích a dále pomocí testovacích vektorů předkládaných komponentám zpracovávající obrazový tok. Jak již bylo popsáno výše, jazyky VHDL a Verilog prakticky neumějí pracovat se složitými soubory typu bitová mapa, JPEG. Načítání obrazových dat do simulací je proto komplikované a testovací obrazce se musejí předpřipravit v jiném vhodném formátu. Další velkou nevýhodou je, že simulace probíhá na úrovni RTL, tedy se provádí funkce na logické úrovni mezi kombinační a sekvenční logikou. Jako příklad uveďme prostý součet dvou 32-bitových čísel A a B s 33-bitovým výsledkem C. V jazyku VHDL bude zkrácený zápis vypadat následovně:

```
Port (
    A, B : in    STD_LOGIC_VECTOR (31 downto 0);
    C    : out   STD_LOGIC_VECTOR (32 downto 0));
end adder;
architecture Behavioral of adder is
begin
  C <= ('0'&A) + ('0'&B);
end Behavioral;
```

Vstupem pro simulaci nebude funkce „+” ale systém stovek devíti-stavových logických funkcí, které simulátor bude postupně vyhodnocovat a zpracovávat. V tomto případě funkční simulace zabere významně více času, než prosté vyhodnocení výrazu s operátorem „+”, jak je tomu např. v jazyku C. Demonstrujme výsledky práce simulátoru na jednoduchém obrazovém filtru s konvoluční maticí 5x5 pracujícím s obrazem o rozměrech 1024 x 1024 obrazových bodů. Výpočet filtru vyžaduje 25 násobení a 24 součtů pro jeden obrazový bod, dohromady 49 operací. Celkem pro celý obraz asi 50 milionů aritmetických operací. Na běžném PC v jazyku C by výpočet trval několik milisekund. Ovšem simulace v RTL bude trvat již několik desítek minut pro jeden předložený vzor. Z výše popsaného vyplývá, že funkční simulace v RTL je nevhodná a zbytečně zdlouhavá.

Rychlejších výsledků lze dosáhnout pomocí speciálních nástrojů tzv. jazyků typu C2H – tedy převodníky jazyka C do HDL jazyků.

V současné době existuje několik jazyku typu C2H. Jedná se například o Handle C, System C a Impulse C. V dalším textu se budeme zabývat pouze jazykem Impulse C, který je velmi podobný jazyku C známého z programovaní mikroprocesoru.

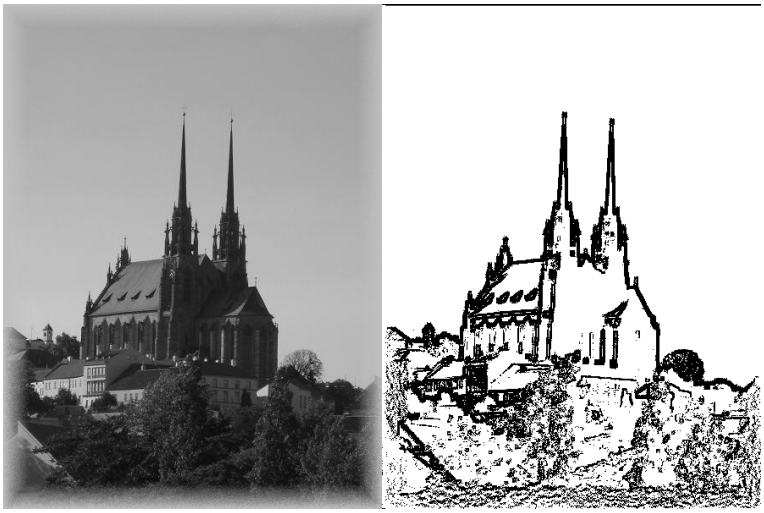
Základním rozhraním je vstupní a výstupní proud dat definovaného typu (integer, UINT_TYPE(XX)...). Dále je datový tok zpracováván pomocí funkcí zapsaných samotným uživatelem v jazyku C, který muže být doplněn řídícími slovy modifikujícími překlad.

Hlavní výhodou řešení je velmi dobrá čitelnost napsaného kódu, krátký čas simulace za použití standardních překladačů (GCC, MSV) umožňující využívat veškeré funkce a knihovny obsažené ve standardním překladači (výpis na obrazovku, breakpointy, čtení z disku). Další výhodou je i rychlé generování HDL kódu pro FPGA. Příklad těla filtru s konvoluční maticí 5x5 je uveden níže. Na první pohled je patrné, jak algoritmus pracuje a je jednoduché ho modifikovat. Využívá vstupního streamu dat (co_stream_read(r0, r1, ...), který načíta nové obrazové body. Do proměnné *sop* je uložena nově vypočtená hodnota obrazového bodu, která je následně zapsána do výstupního datového streamu. Z programu je vidět, že programátor nemusí primárně řešit paralelizovaní zpracovávané úlohy. O tom se rozhoduje v překladači pomocí řídících slov a parametrů nastavení překladu.

```
do {
#pragma CO PIPELINE
#pragma CO set stageDelay 100
      err  = co_stream_read(r0, &data0, sizeof(co_uint16));
      err &= co_stream_read(r1, &data1, sizeof(co_uint16));
      err &= co_stream_read(r2, &data2, sizeof(co_uint16));
      err &= co_stream_read(r3, &data3, sizeof(co_uint16));
      err &= co_stream_read(r4, &data4, sizeof(co_uint16));
      if (err != co_err_none) break;

      p00 = p01; p01 = p02; p02 = p03; p03 = p04;
      p10 = p11; p11 = p12; p12 = p13; p13 = p14;
      p20 = p21; p21 = p22; p22 = p23; p23 = p24;
      p30 = p31; p31 = p32; p32 = p33; p33 = p34;
      p40 = p41; p41 = p42; p42 = p43; p43 = p44;

      p04 = data0;
      p14 = data1;
      p24 = data2;
      p34 = data3;
      p44 = data4;

      sop = p00*F00 + p01*F01 + p02*F02 + p03*F03 + p04*F04
         + p10*F10 + p11*F11 + p12*F12 + p13*F13 + p14*F14
         + p20*F20 + p21*F21 + p22*F22 + p23*F23 + p24*F24
         + p30*F30 + p31*F31 + p32*F32 + p33*F33 + p34*F34
         + p40*F40 + p41*F41 + p42*F42 + p43*F43 + p44*F44;
      if (sop > 255*FDIV)
         result = 255;
      else
         result = (co_uint16) (sop >> 7);  // Divide by 128
      co_stream_write(output_stream, &result, sizeof(co_uint16));
   } while (1);
   co_stream_close(r0);
   co_stream_close(r1);
   co_stream_close(r2);
   co_stream_close(r3);
   co_stream_close(r4);
   co_stream_close(output_stream);
}
```

Vstupní a výstupní obraz získaný výpočtem funkce filtru je na obr. 3.

Dále je výsledný HDL kód připojen jako předgenerovaná komponenta do standardního řetězce FPGA nástrojů, kde může být propojena s ostatními systémovými komponentami. Jednou ze systémových komponent může být soft-core procesor Microblaze nebo hard-core procesor PowerPC sloužící pro další zpracování informací.



Obr. 2 Vstupní a výstupní obrázek filtru

3. Příklad praktické aplikace

Aplikace, která vyžadovala netradiční přístup k řešení, byl návrh rychloběžné kamery s vyhodnocením těžiště stopy laserového paprsku na projekční stěně zbraňového simulátoru.

Primárním cílem projektu bylo navrhnout a realizovat kamerový systém schopný detekovat a vyhodnotit pozici na kterou míří až 10 střelců se vzorkovací frekvencí 50Hz na jednu zbraň s přesností 1mm při uvažované velikosti snímané scény 1,2 m x 1 m. Každá zbraň je vybavena spínanou IR laserovou diodou, která po průchodu optickou soustavou vytváří stopu na projekční ploše.

Jednotlivé zbraně jsou rozděleny do časových multiplexů a to v poměru 1 : 10 na každou zbraň. Aby bylo dosaženo vyšší přesnosti, je ještě každý časový usek rozdělen na dalších 50 časových úseků. Tedy každých 20 ms jsou vyhodnoceny pozice všech zabraní a dále se tento cyklus periodicky opakuje 50 krát za 1s. Celková vzorkovací frekvence a tím i snímková frekvence kamery musí být 500 snímků za 1s.

Druhým požadavkem je minimální velikost (1mm) obrazového bodu, který má být detekován. Vezmeme-li do úvahy velikost snímané scény, je nezbytně nutné, aby snímací element měl rozlišení minimálně 1200 x 1000 obrazových bodů.

Výše zmíněný popis jasně definuje technické požadavky na kameru i vyhodnocovací systém.

Následně bylo třeba vybrat systém, který zpracuje a vyhodnotí datový tok pořízený vybraným CMOS čipem. Rychlým výpočtem dojdeme k závěru, že vyhodnocovací systém musí byt schopen zpracovat asi 780 Mbytů dat za 1s a to ještě v nezarovnaném 10bitovém formátu. Doba potřebná pro vyhodnocení jednoho obrazového bodu je asi ***1,5 ns***.

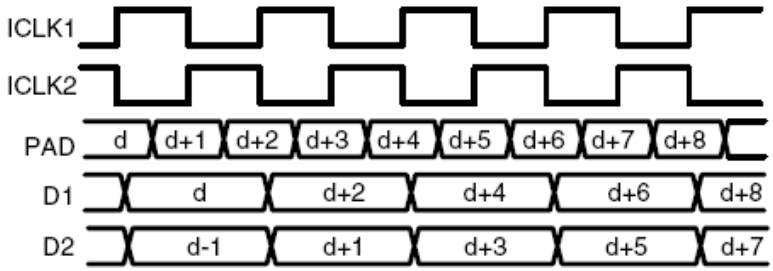
Výše zmíněné požadavky lze řešit využitím architektury založené na hradlových polích typu FPGA, které disponují vhodným rozhraním a dostatečnou mohutností logiky uvnitř čipu.

V první části příspěvku bylo již řečeno, že data jsou z čipu získávána rychlosti 780MB za 1s. Je logické, že datový tok z čipu bude rozdělen do více pracovních kanálů. Tak je tomu i v případě CMOS čipu firmy Cypress. Datový interface je rozdělen na hodinový signál, který určuje rychlost vyčítaní obrazových dat na ostatních kanálech, řídící kanál, který generuje informace o stavu CMOS čipu (začátek, konec snímku, začátek konec řádku, platná data, testovací vzor a kontrolní součet) a dvanáct datových kanálů. Každý kanál je tvořen fyzickou vrstvou diferenciálního rozhraní LVDS pracující na dvojnásobné frekvenci než základní hodinový signál (na každou hranu hodinového signálu přichází jeden datový bit).

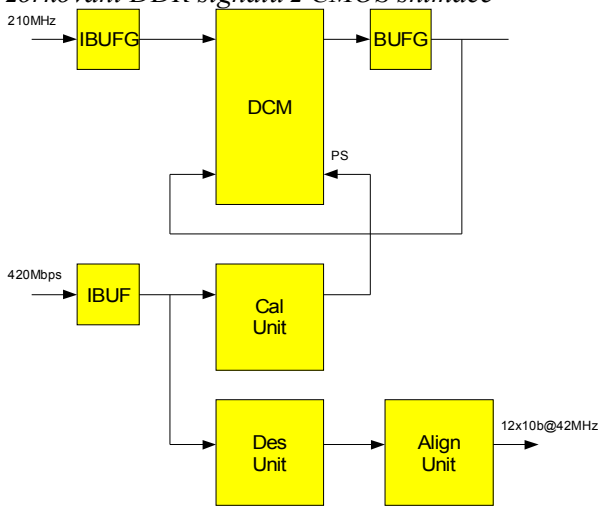
Výběr hradlového pole předurčuje definice interfacu a požadavků na zpracování získaných dat z kamery. V průběhu návrhu kamery bylo vybíráno ze tří typů hradlových polí firmy Xilinx. Jednalo se o řady Spartan3E, Virtex 4 a Virtex 5. Vlastnosti jednotlivých rodin nebudou dále v článku diskutovány, protože by překročili rozsah příspěvku viz. [1], ale ve stručnosti lze zmínit, že rodiny Virtex poskytují větší výkon a komfort při vlastním návrhu ale na druhé straně zvyšují výslednou cenu zapojení. Rodiny Spartan3E jsou levnější s omezenou funkcionalitou, která ještě na samotné hranici technických možností vyhovuje požadavkům cílové aplikace.

Rozhraní pro získávání dat musí byt navrženo s ohledem na vysokou přenosovou rychlost komunikačního kanálu – v našem případě 620Mbps a to ve 12 datových kanálech a jednom kanále synchronizačním.

Dalším problém byl neznámý vztah mezi fázi hodinového signálu a datových signálů v ostatních kanálech. Optimálních poměrů při vzorkování bylo dosaženo pomocí automatické kalibrační jednotky, která byla za tímto účelem vytvořena a využívala k fázovým posunům jednotku DCM integrovanou v FPGA Spartan3E. Automatické kalibrace probíhala pomocí nastavení definovaného vzoru o nejvyšší možné generované frekvenci (vzor typu 0101010101), kdy stavový automat měnil fázový posun tak, aby našel konstantní vzdálenost mezi náběžnou a sestupnou hranou.



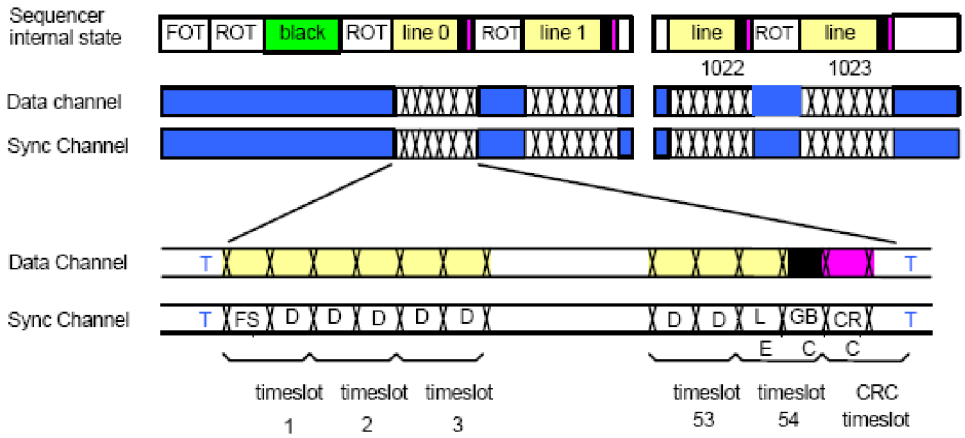
Obr. 3 Vzorkování DDR signálů z CMOS snímače



Obr. 4 Vnitřní uspořádání kalibrační a zarovnávací jednotky

Druhá synchronizace probíhala již na úrovni sestavení celého 10-ti bitového slova a to ve formě jiného testovacího vzoru, který se nemůže vyskytnout v synchronizačním kanále za bezporuchového provozu. Jakmile byl předpokládaný synchronizační vzor zachycen, došlo k ověření synchronizace i v ostatních kanálech. V případě, že všechny kanály byly synchronizovány je možno považovat data za platná ve všech dvanácti datových kanálech a i v synchronizačním kanálu. Výsledným produktem jsou deserializovaná data a řídící

informace na jedné pětině základní frekvence ve 12 + 1 datových proudech (základní frekvence hodin 310MHz, DDR = 620Mbit, tedy 62MHz slov za 1s v každém kanálu).



Obr. 5 Časové posloupnosti dat v synch. a datovém kanálu

Získaný datový tok je třeba dále upravit a přeuspořádat do pořadí vhodného k vyhodnocení, protože data z komunikačních kanálů vystupují dle algoritmu udaného výrobcem čipu (12 x sudé pixely, 12 x liché, 12 liché v reverzním pořadí a 12 sudé v reverzním pořadí).

Po finálním přeuspořádání je možné přistoupit k vlastnímu procesu vyhodnocení obrazových dat a získání těžiště stopy laserového paprsku. Řešení systému využívá masivního paralelismu architektury založené na struktuře FPGA, kdy každý datový proud je zpracováván samostatně stejným algoritmem a výsledky zpracování postupují pipeline jednotkou k dalším jednotkám zajištujícím požadované výsledky.

V první části je vhodné odprahovat pozadí z projekční stěny od aktivní stopy laserového paprsku. To probíhá ve dvou fázích. První fáze je oddělení IR spektra laserového paprsku pomocí IR filtru, který je před objektivem kamery. Tím je zajištěno, že do kamery a tedy i do vyhodnocovacího systému se dostane převážně účinný signál IR laseru.

Ve druhé fázi se hledá vhodný práh, který oddělí zbytkové pozadí od IR stopy laseru a v okně o rozměrech 8 x 8 obrazových bodů se provádí výpočet těžiště bodu dle vztahů (1). Pro jednoduchost jsou souřadnice počítány v relativních souřadnicích a tak postačuje k získání validních dat 18bitová aritmetika.

$$x_T = \sum_i \frac{m_i x_i}{m} \qquad y_T = \sum_i \frac{m_i y_i}{m} \qquad m = \sum_i m_i \tag{1}$$

V průběhu řešení projektu se nepředpokládalo, že by stopa paprsku přesáhla velikost 4 x 4 obrazové body. Systém je schopen vyhodnotit až 10 nepřekrývajících se oblastí i když při správném nastavení a podmínkách provozu by se na projekční stěně měla vyskytovat pouze jedna platná stopa.

Systém byl dále doplněn o video výstup na VGA monitor v rozlišení 1280 x 1024 obrazových bodů pro kalibraci a ověřování funkce vyhodnocovací jednotky. Zde je třeba poznamenat, že bylo nezbytně nutné snížit snímkovou frekvenci na 60Hz z důvodu přímého zobrazení na monitoru, protože systém není vybaven žádnou vyrovnávací pamětí, do které by bylo možno ukládat získanou obrazovou informaci a následně ji zobrazovat nebo předávat do jiného zařízení.

Předávání výsledků získaných z kamery je realizováno pomocí sériového portu nebo rozhraní 100Mbit Ethernet. Doplňkové ovládání funkcí čipu je zajištěno pomocí SPI linky, které je rovněž vytvořena jako IP funkce v hradlovém poli. Přehled dosažených parametrů je v tabulce 2.

Tabulka 2: Dosazené parametry systému

Parametr	Hodnota	Jednotky
Frekvence hodin čipu	40 – 320	MHz
Bitový tok na kanál	80 – 640	Mbit/s
Počet zpracovaných slov	96 – 768	Mslov/s
Snímková frekvence	50 – 500	Hz
Počet vyhodnocených pozic při 10 zbraních	5 – 50	1/s
Příkon systému @ 500Hz	6	W

Obr. 6 Pohled na scénu a použité zbraně

ZÁVĚR

Cílem článku bylo přiblížit zpracování obrazových dat architekturou FPGA a obeznámit čtenáře s úskalími, na které muže v průběhu řešení projektu narazit. Ve druhé části byla uvedena aplikace reálného systému rychloběžné kamery využívající popsaná řešení pro vyhodnocení těžiště stopy laserového paprsku na projekční stěně zbraňového simulátoru. Lze se oprávněně domnívat, že technologie automatického a poloautomatického převodu softwarových řešení do hardwarových struktur se bude nadále vyvíjet, a proto by ji měla být věnována nemalá pozornost.

PODĚKOVÁNÍ

Článek vznikl za podpory projektu 1M0567 (Výzkumné centrum aplikované kybernetiky).

LITERATURA

[1] Stránky společnosti Xilinx, internetový odkaz: http://www.xilinx.com/

[2] Stránky společnosti Cypress, internetový odkaz: http://www.cypress.com/

[3] Stránky společnosti Impulse, internetový odkaz: http://www.impulsec.com/